Leon

Minh Chau Do

09 10 2023

Daten von Leon

```
#Datensortierung

# In Mortalität umrechnen
leon_r$V2 <- 100 - leon_r$V2

leon_r <- leon_r[complete.cases(leon_r),]

leon_r <- leon_r[order(leon_r$V2), ]

# Werte außerhalb des Bereichs [0, inf] raus
leon_r <- leon_r[leon_r$V1 >= 0, ]
xr_leon <- sort(leon_r$V1)
yr_leon <- leon_r$V2



# In Mortalität umrechnen
leon_b$V2 <- 100 - leon_b$V2

leon_b <- leon_b[complete.cases(leon_b),]

leon_b <- leon_b[order(leon_b$V2), ]

leon_b <- leon_b[leon_b$V1 >= 0, ]
xb_leon <- sort(leon_b$V1)
yb_leon <- leon_b$V2
```

Startfunktion

Um die bestmögliche Anpassung an den vorliegenden Datenpunkten darzustellen wurde mithilfe einer Funktion der beste Startparameter mit dem kleinsten Fehler errmittelt, da die Anpassung stark von den initialen Startwerten abhängig ist.

```
find_best_start_2parameter <- function(p, q, max_beta, max_eta, steps_beta,
                                       steps_eta, fitting_function) {

  best_errors <- numeric()  # Vektor für Fehlerwerte
  best_starts <- matrix(nrow = 0, ncol = 2)  # Matrix für Startparameter
```

```
  for (beta in seq(0, max_beta, by = steps_beta)) {
    for (eta in seq(0, max_eta, by = steps_eta)) {
      start_params <- c(beta, eta)

      # Schätze die Parameter mit den aktuellen Startparametern
      if(identical(as.character(substitute(fitting_function)), "getstuexp2")){
        output <- capture.output({
        result <- getstuexp2(
                        p = p, q = q, start = start_params,
                        show.output = TRUE, plot = FALSE, wert1 = 2
                      )
        })
      }
      else{
        output <- capture.output({
        result <- fitting_function(p = p, q = q, start = start_params,
                              show.output = TRUE, plot = FALSE)
        })
      }

      # Berechne den Fehler (leon_r_1$value) für die aktuellen Startparameter
      current_error <- as.numeric(gsub("\\[1\\]\\s+", "", output[5]))

      # Speichere Fehler und die Startparameter, wenn der Fehler nicht NA ist
      if (!is.na(current_error)) {
        best_errors <- c(best_errors, current_error)
        best_starts <- rbind(best_starts, start_params)
      }
    }
  }


  # Finde den Index des kleinsten Fehlers (ignoriere NA-Werte)
  best_index <- which.min(best_errors)

  # Wähle den besten Startparameter mit dem kleinsten Fehler aus
  best_start <- best_starts[best_index, ]

  # Gib den besten Startparameter und den entsprechenden Fehler aus
  cat("Bester Startparameter:", best_start, "\n")
  cat("Bester Fehler:", best_errors[best_index], "\n")

  return(best_start)
}


find_best_start_3parameter <- function(p, q, max_shape1 = 10, max_shape2 = 10,
                                       max_scale = 10, steps_shape1, steps_shape2,
                                       steps_scale, fitting_function) {

  best_errors <- numeric()  # Vektor für Fehlerwerte
  best_starts <- matrix(nrow = 0, ncol = 3)  # Matrix für Startparameter
```

```r
  for (shape1 in seq(0, max_shape1, by = steps_shape1)) {
    for (shape2 in seq(0, max_shape2, by = steps_shape2)) {
      for (scale in seq(0, max_shape1, by = steps_scale)) {
        start_params <- c(shape1, shape2, scale)

        # Schätze die Parameter mit den aktuellen Startparametern
        if(identical(as.character(substitute(fitting_function)), "getstuexp3")){
          output <- capture.output({
            result <- getstuexp3(
                          p = p, q = q, start = start_params,
                          show.output = TRUE, plot = FALSE, wert1 = 2, wert2 = 6)
          })
      }

        else{
          output <- capture.output({
            result <- fitting_function(p = p, q = q, start = start_params,
                                show.output = TRUE, plot = FALSE)
          })
        }
        # Berechne den Fehler (leon_r_1$value) für die aktuellen Startparameter
        current_error <- as.numeric(gsub("\\[1\\]\\s+", "", output[5]))

        # Speichere Fehler und die Startparameter, wenn der Fehler nicht NA ist
        if (!is.na(current_error)) {
          best_errors <- c(best_errors, current_error)
          best_starts <- rbind(best_starts, start_params)
        }
      }
    }
  }

  # Finde den Index des kleinsten Fehlers (ignoriere NA-Werte)
  best_index <- which.min(best_errors)

  # Wähle den besten Startparameter mit dem kleinsten Fehler aus
  best_start <- best_starts[best_index, ]

  # Gib den besten Startparameter und den entsprechenden Fehler aus
  cat("Bester Startparameter:", best_start, "\n")
  cat("Bester Fehler:", best_errors[best_index], "\n")

  return(best_start)
}




find_best_start_4parameter <- function(p, q, max_shape, max_scale, max_rate,
                                       max_mix, steps_shape, steps_scale,
                                       steps_rate, steps_mix,fitting_function) {

  best_errors <- numeric()  # Vektor für Fehlerwerte
```

```
  best_starts <- matrix(nrow = 0, ncol = 4)  # Matrix für Startparameter

  for (shape in seq(0, max_shape, by = steps_shape)) {
    for (scale in seq(0, max_scale, by = steps_scale)) {
      for (rate in seq(0, max_rate, by = steps_rate)) {
        for (mix in seq(0, max_mix, by = steps_mix)) {
          start_params <- c(shape, scale, rate, mix)

          # Schätze die Weibull-Parameter mit den aktuellen Startparametern
          output <- capture.output({
            result <- fitting_function(p = p, q = q, start = start_params,
                                       show.output = TRUE, plot = FALSE)
          })

          # Berechne den Fehler (leon_r_1$value) für die aktuellen Startparameter
          current_error <- as.numeric(gsub("\\[1\\]\\s+", "", output[5]))

          # Speichere Fehler und die Startparameter, wenn der Fehler nicht NA ist
          if (!is.na(current_error)) {
            best_errors <- c(best_errors, current_error)
            best_starts <- rbind(best_starts, start_params)
          }
        }
      }
    }
  }

  # Finde den Index des kleinsten Fehlers (ignoriere NA-Werte)
  best_index <- which.min(best_errors)

  # Wähle den besten Startparameter mit dem kleinsten Fehler aus
  best_start <- best_starts[best_index, ]

  # Gib den besten Startparameter und den entsprechenden Fehler aus
  cat("Bester Startparameter:", best_start, "\n")
  cat("Bester Fehler:", best_errors[best_index], "\n")

  return(best_start)
}
```

Datenanpassung an die Daten von Leon

Weibullverteilung

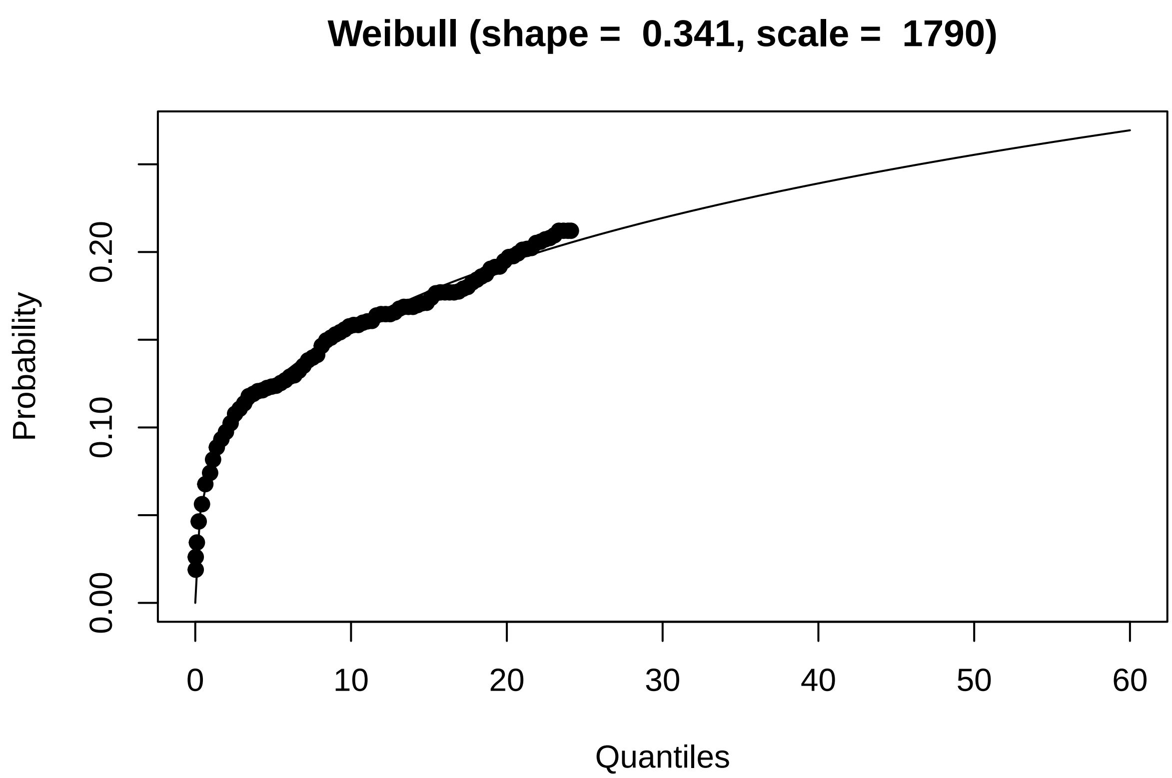
```
# Weibullverteilung
source("C:/Users/Chau/Documents/Masterarbeit/R Funktionen/getweibullpar.R")

best_leon_r_1 <- find_best_start_2parameter(p = yr_leon/100, q = xr_leon,
                                            max_beta = 10, max_eta = 10,
                                            steps_beta = 1, steps_eta = 1,
                                            fitting_function = getweibullpar)
```

```
## Bester Startparameter: 3 1
```

```
## Bester Fehler: 1.993567e-07
```

```
leon_r_1 <- getweibullpar(
                        p = yr_leon/100,
                        q = xr_leon,
                        start = best_leon_r_1,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1]    0.3414611 1786.4564803
##
## $value
## [1] 1.993567e-07
##
## $counts
## function gradient
##       50       50
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

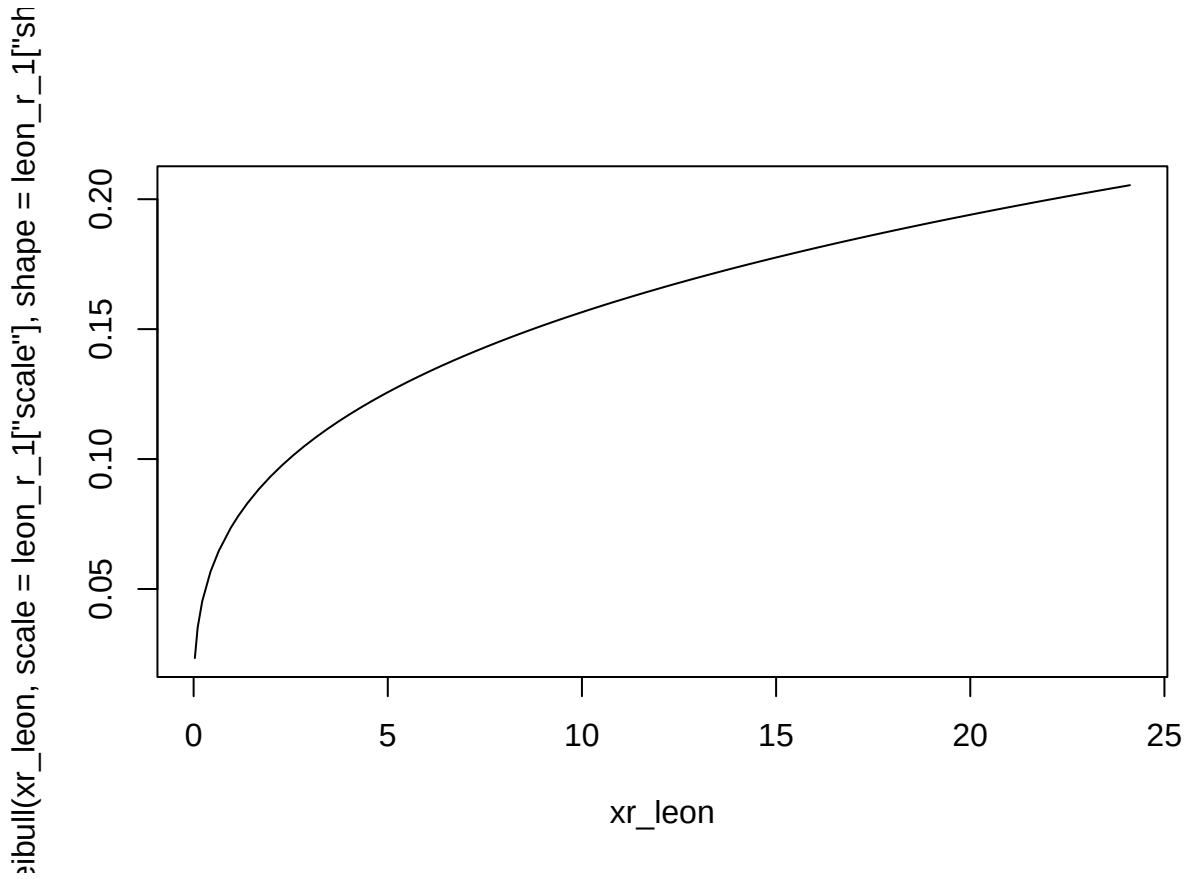


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
leon_r_1
```

```
##        shape        scale
##    0.3414611 1786.4564803
```

```
plot(xr_leon,
     pweibull(xr_leon,
              scale = leon_r_1["scale"],
              shape = leon_r_1["shape"]), type = "l")
```



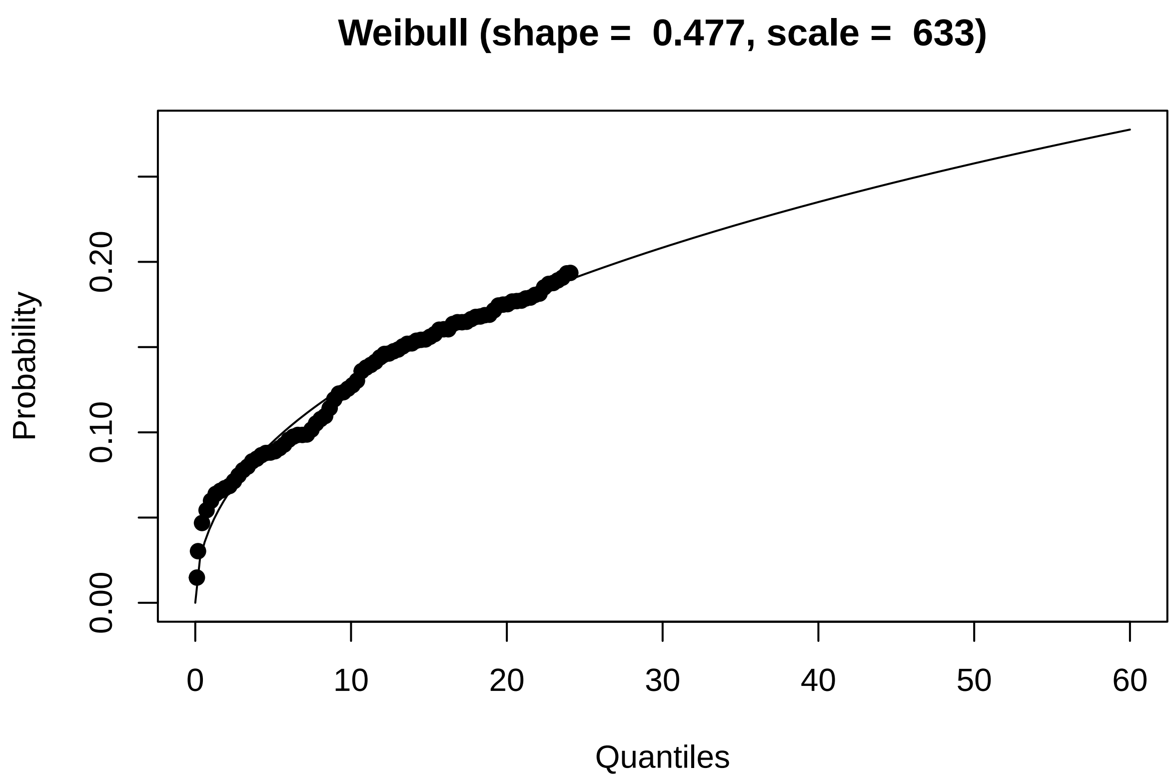

```
best_leon_b_1 <- find_best_start_2parameter(p = yb_leon/100, q = xb_leon,
                                            max_beta = 10, max_eta = 10,
                                            steps_beta = 1, steps_eta = 1,
                                            fitting_function = getweibullpar)
```

```
## Bester Startparameter: 2 1
## Bester Fehler: 3.652618e-07
```

```
leon_b_1 <- getweibullpar(
                        p = yb_leon/100,
                        q = xb_leon,
                        start = best_leon_b_1,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

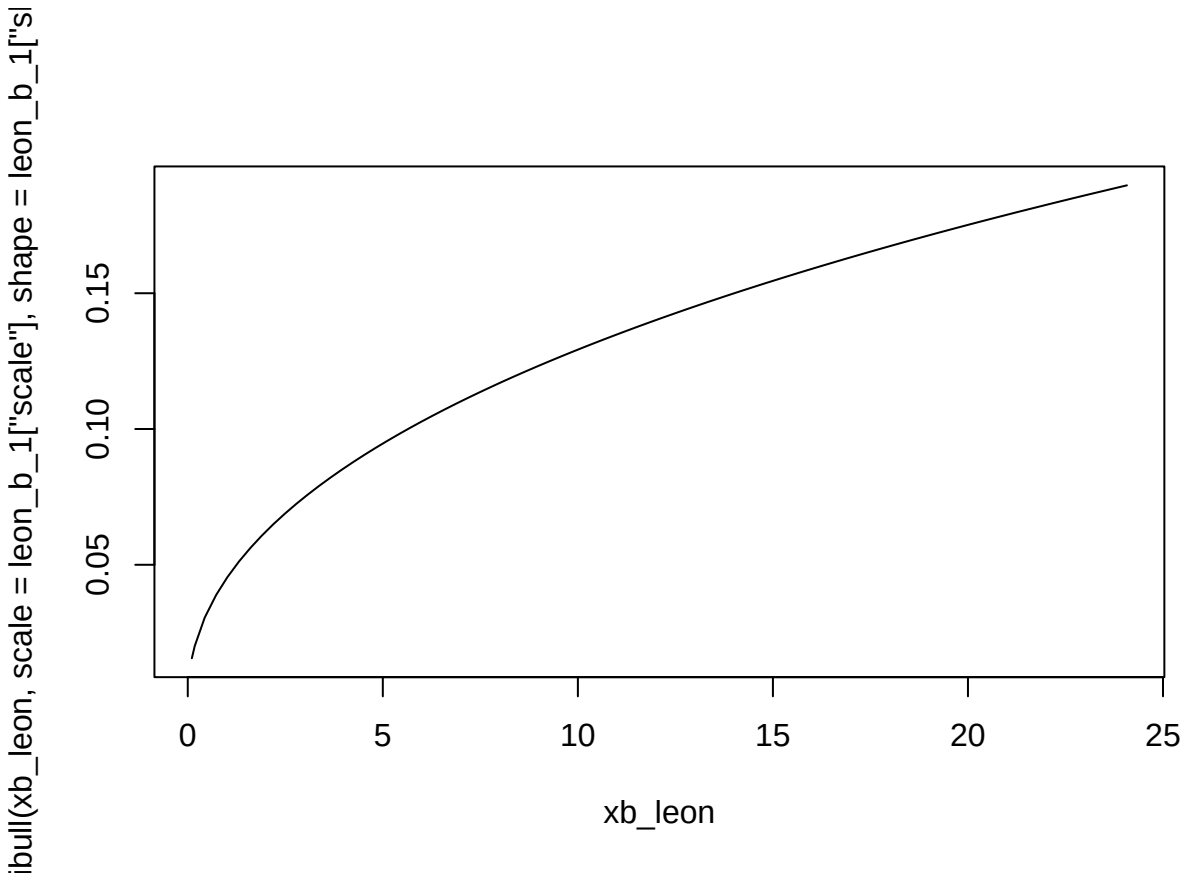
```
## $par
## [1]   0.4768213 632.9334399
```

```
##
## $value
## [1] 3.652618e-07
##
## $counts
## function gradient
##       52       52
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```



., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

```
plot(xb_leon,
     pweibull(xb_leon,
              scale = leon_b_1["scale"],
              shape = leon_b_1["shape"]), type = "l")
```



Exponentiierte Weibullverteilung

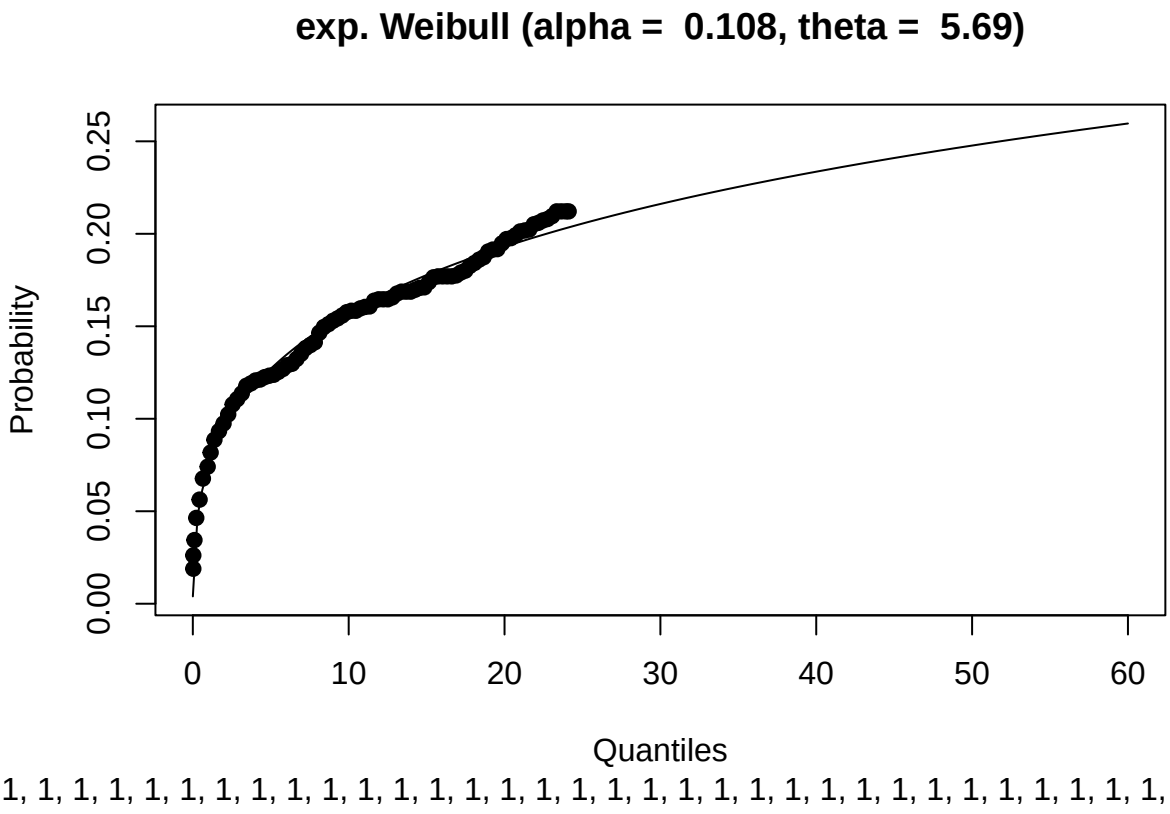
```
# exponentiierte Weibullverteilung
source("C:/Users/Chau/Documents/Masterarbeit/R Funktionen/getweibpar.R")

best_weibbull_xr_leon <- find_best_start_2parameter(p = yr_leon/100,
                                                    q = xr_leon,
                                                    max_beta = 10,
                                                    max_eta = 10,
                                                    steps_beta = 1,
                                                    steps_eta = 1,
                                                    fitting_function = getweibpar)
```

```
## Bester Startparameter: 4 1
## Bester Fehler: 2.589699e-07
```

```
weibbull_xr_leon <- getweibpar(
                        p = yr_leon/100,
                        q = xr_leon,
                        start = best_weibbull_xr_leon,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 0.1078726 5.6862935
```

```
##
## $value
## [1] 2.589699e-07
##
## $counts
## function gradient
##       16       16
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

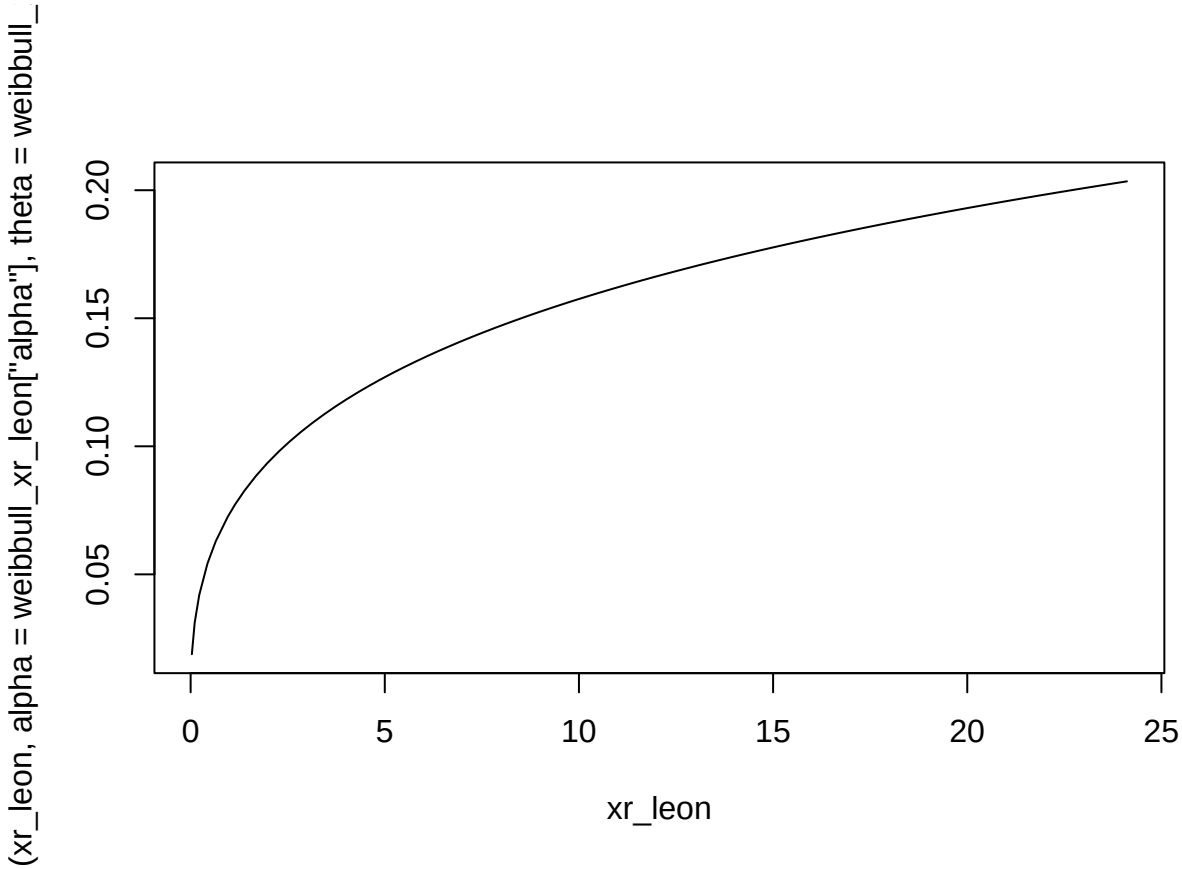


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
weibbull_xr_leon
```

```
##     alpha     theta
## 0.1078726 5.6862935
```

```
plot(xr_leon,
     pexpo.weibull(xr_leon,
               alpha = weibbull_xr_leon ["alpha"],
               theta = weibbull_xr_leon ["theta"]), type = "l")
```



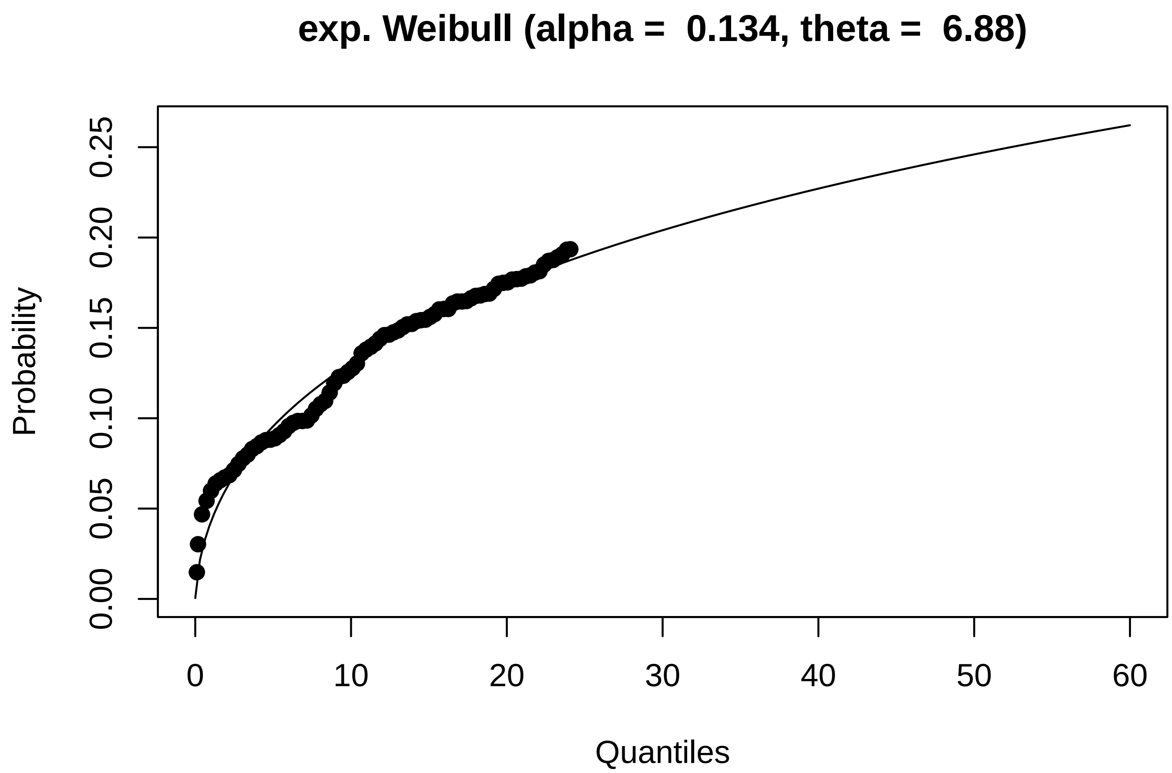

```
best_weibbull_xb_leon <- find_best_start_2parameter(p = yb_leon/100,
                                                    q = xb_leon,
                                                    max_beta = 10,
                                                    max_eta = 10,
                                                    steps_beta = 1,
                                                    steps_eta = 1,
                                                    fitting_function = getweibpar)
```

```
## Bester Startparameter: 5 0
## Bester Fehler: 4.993476e-07
```

```
weibbull_xb_leon  <- getweibpar(
                        p = yb_leon/100,
                        q = xb_leon,
                        start = best_weibbull_xb_leon,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 0.1343066 6.8841315
##
## $value
## [1] 4.993476e-07
##
## $counts
## function gradient
```

```
##        31       31
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

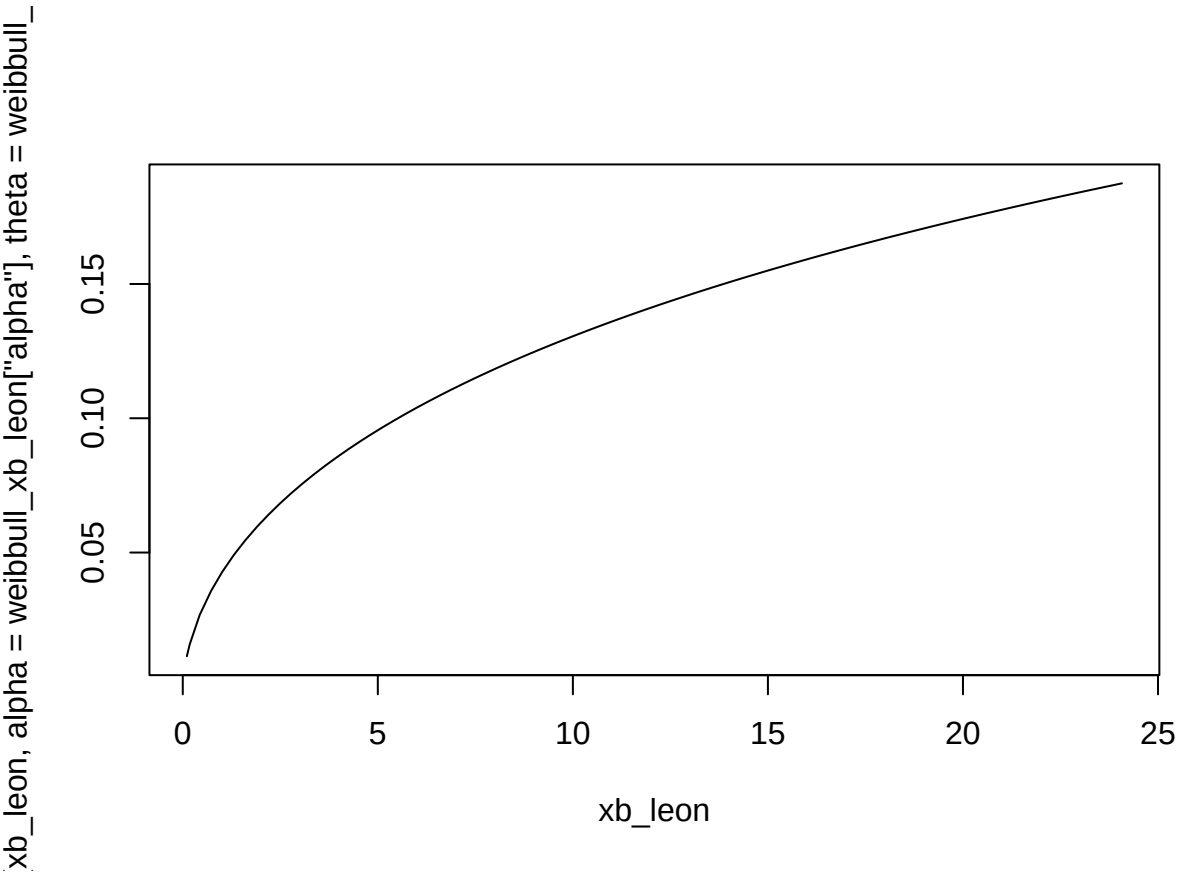


., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

```
weibbull_xb_leon
```

```
##     alpha     theta
## 0.1343066 6.8841315
```

```
plot(xb_leon,
     pexpo.weibull(xb_leon,
              alpha = weibbull_xb_leon ["alpha"],
              theta = weibbull_xb_leon ["theta"]), type = "l")
```



Mischung aus Weibull- und Exponentialverteilung

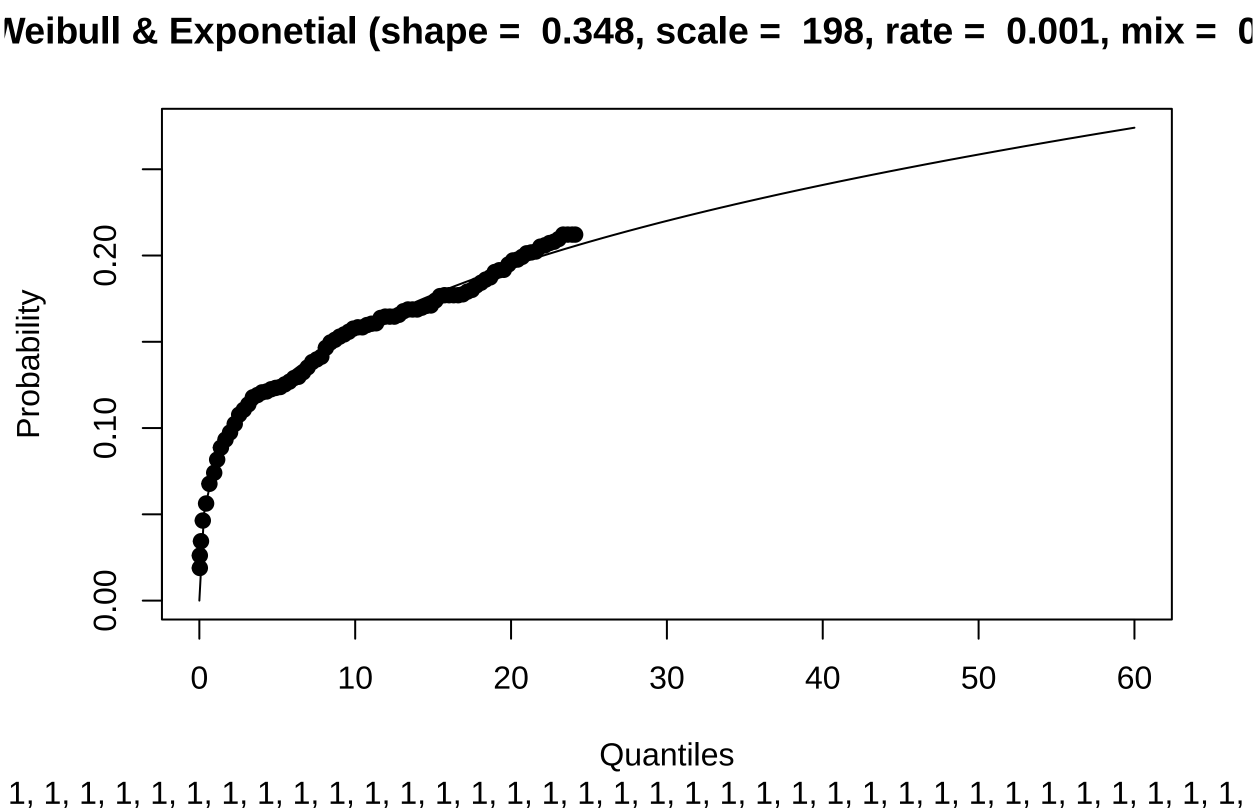
```
# Mischung aus Weibull- und Exponentialverteilung
source("C:/Users/Chau/Documents/Masterarbeit/R Funktionen/getweibex.R")

best_weibex_xr_leon <- find_best_start_4parameter(p = yr_leon/100,
                                                  q = xr_leon,
                                                  max_shape = 1,
                                                  max_scale = 100,
                                                  max_rate = 0.7,
                                                  max_mix = 1,
                                                  steps_shape = 0.1,
                                                  steps_scale = 20,
                                                  steps_rate = 0.1,
                                                  steps_mix = 0.1,
                                                  fitting_function = getweibex)
```

```
## Bester Startparameter: 0.9 0 0.3 0.7
## Bester Fehler: 1.869176e-07
```

```
weibex_xr_leon <- getweibex(
                        p = yr_leon/100,
                        q = xr_leon,
                        start = best_weibex_xr_leon, # c(0.5, 60, 0.4, 0.1),
                        show.output = TRUE,
                        plot = TRUE
```

```
                      )
```

```
## $par
## [1]   0.347918 197.750556   0.001000   0.030844
##
## $value
## [1] 1.869176e-07
##
## $counts
## function gradient
##       40       40
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

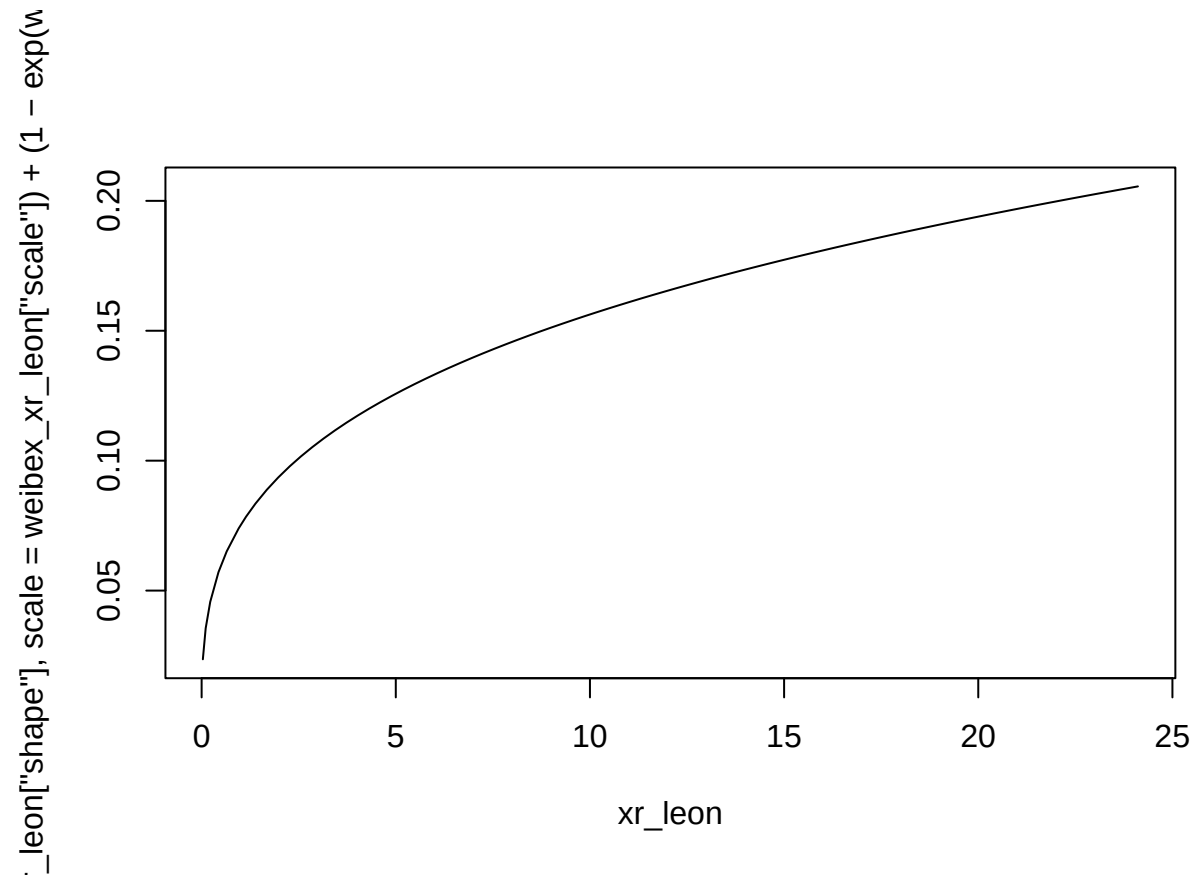



```
weibex_xr_leon
```

```
##      shape      scale       rate        mix
##   0.347918 197.750556   0.001000   0.030844
```

```
# Wenn Ergebnisse aus weibex_xr1_1 von Funktion abgelesen
plot(xr_leon,
     (exp(weibex_xr_leon["mix"]) / ( 1 + exp(weibex_xr_leon["mix"])) *
        stats::pweibull(q = xr_leon,
```

```
                        shape = weibex_xr_leon["shape"],
                        scale = weibex_xr_leon["scale"]) +
       (1 - exp(weibex_xr_leon["mix"]) / ( 1 + exp(weibex_xr_leon["mix"]))) *
       stats::pexp(q = xr_leon,
                   rate = weibex_xr_leon["rate"])),
     type = "l")
```



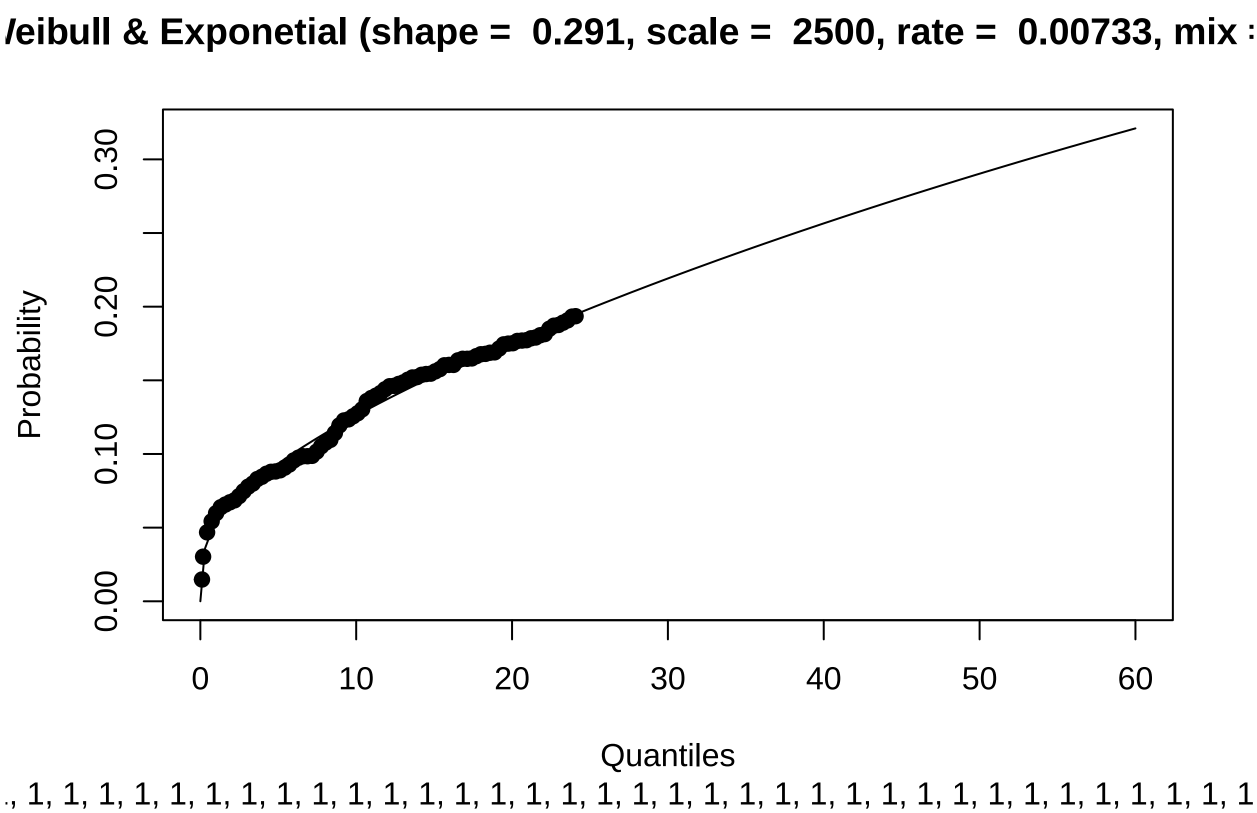

```
best_weibex_xb_leon <- find_best_start_4parameter(p = yb_leon/100,
                                                  q = xb_leon,
                                                  max_shape = 1,
                                                  max_scale = 100,
                                                  max_rate = 0.7,
                                                  max_mix = 1,
                                                  steps_shape = 0.1,
                                                  steps_scale = 20,
                                                  steps_rate = 0.1,
                                                  steps_mix = 0.1,
                                                  fitting_function = getweibex)
```

```
## Bester Startparameter: 0.5 0 0 0.1
## Bester Fehler: 2.427772e-07
```

```
weibex_xb_leon <- getweibex(
                        p = yb_leon/100,
                        q = xb_leon,
                        start = best_weibex_xb_leon, # c(0.5, 60, 0.7, 0.1),
                        show.output = TRUE,
```

```
                        plot = TRUE
                      )
```

```
## $par
## [1] 2.914738e-01 2.495124e+03 7.327208e-03 1.087174e-03
##
## $value
## [1] 2.427772e-07
##
## $counts
## function gradient
##       29       29
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

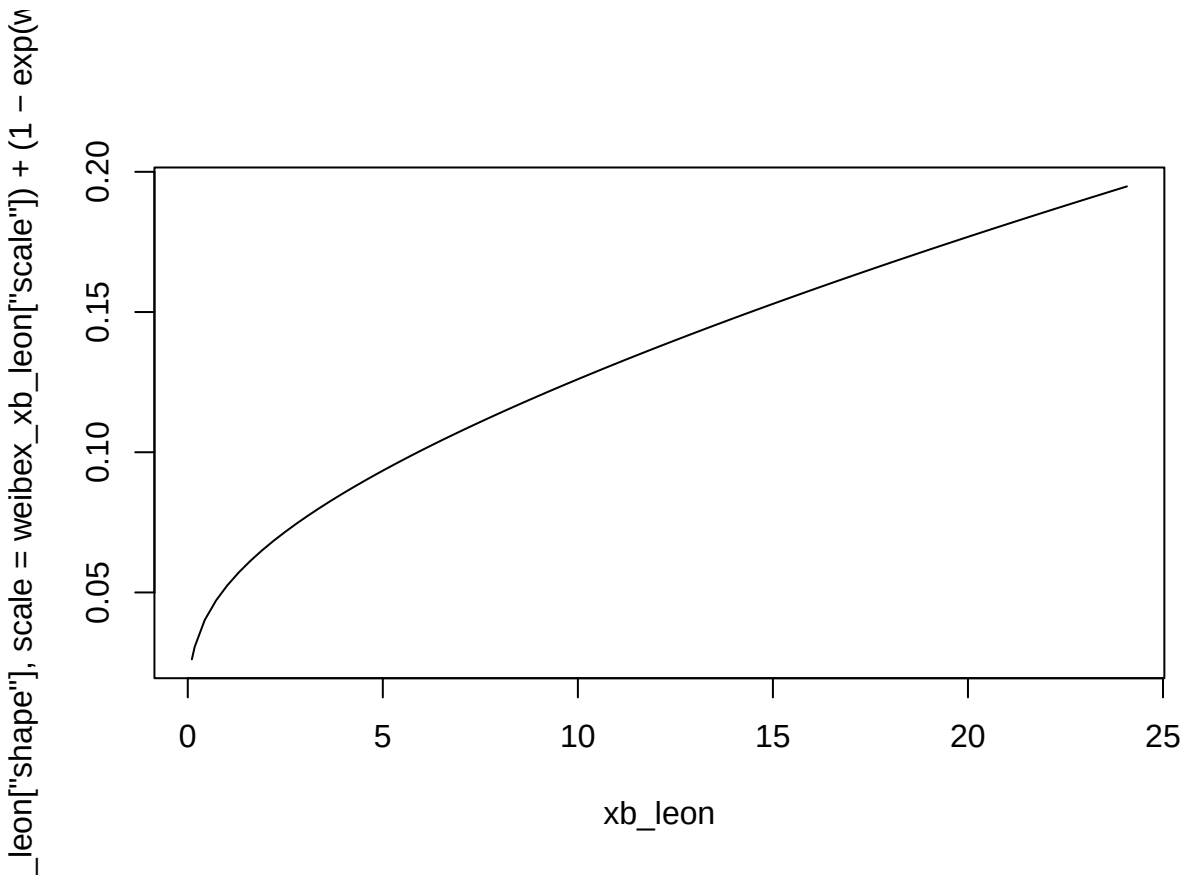



```
weibex_xb_leon
```

```
##        shape        scale         rate          mix
## 2.914738e-01 2.495124e+03 7.327208e-03 1.087174e-03
```

```
plot(xb_leon,
     (exp(weibex_xb_leon["mix"]) / ( 1 + exp(weibex_xb_leon["mix"])) *
        stats::pweibull(q = xb_leon,
```

```
                        shape = weibex_xb_leon["shape"],
                        scale = weibex_xb_leon["scale"]) +
       (1 - exp(weibex_xb_leon["mix"]) / ( 1 + exp(weibex_xb_leon["mix"]))) *
       stats::pexp(q = xb_leon,
                   rate = weibex_xb_leon["rate"])),
     type = "l")
```



Mischung von exponentiierter Weibull- und Exponentialverteilung

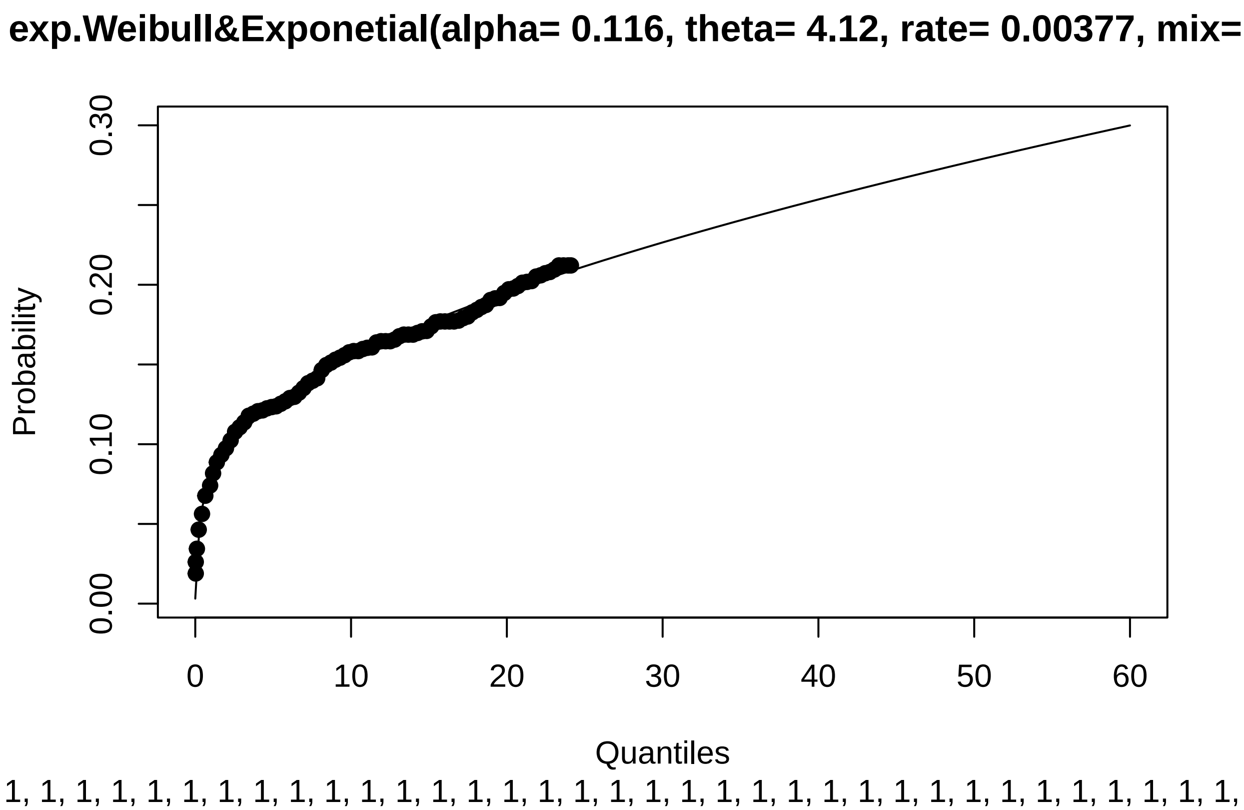
```
# Mischung von exponentiierter Weibull- und Exponentialverteilung
source("C:/Users/Chau/Documents/Masterarbeit/R Funktionen/get2weibex.R")

best_weibex2_xr_leon <- find_best_start_4parameter(p = yr_leon/100,
                                                   q = xr_leon,
                                                   max_shape = 1,
                                                   max_scale = 100,
                                                   max_rate = 0.7,
                                                   max_mix = 1,
                                                   steps_shape = 0.1,
                                                   steps_scale = 20,
                                                   steps_rate = 0.1,
                                                   steps_mix = 0.1,
                                                   fitting_function = get2weibex)
```

```
## Bester Startparameter: 0.8 0 0.5 0.2
```

```
## Bester Fehler: 1.350252e-07
```

```
weibex2_xr_leon <- get2weibex(
                        p = yr_leon/100,
                        q = xr_leon,
                        start = best_weibex2_xr_leon,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

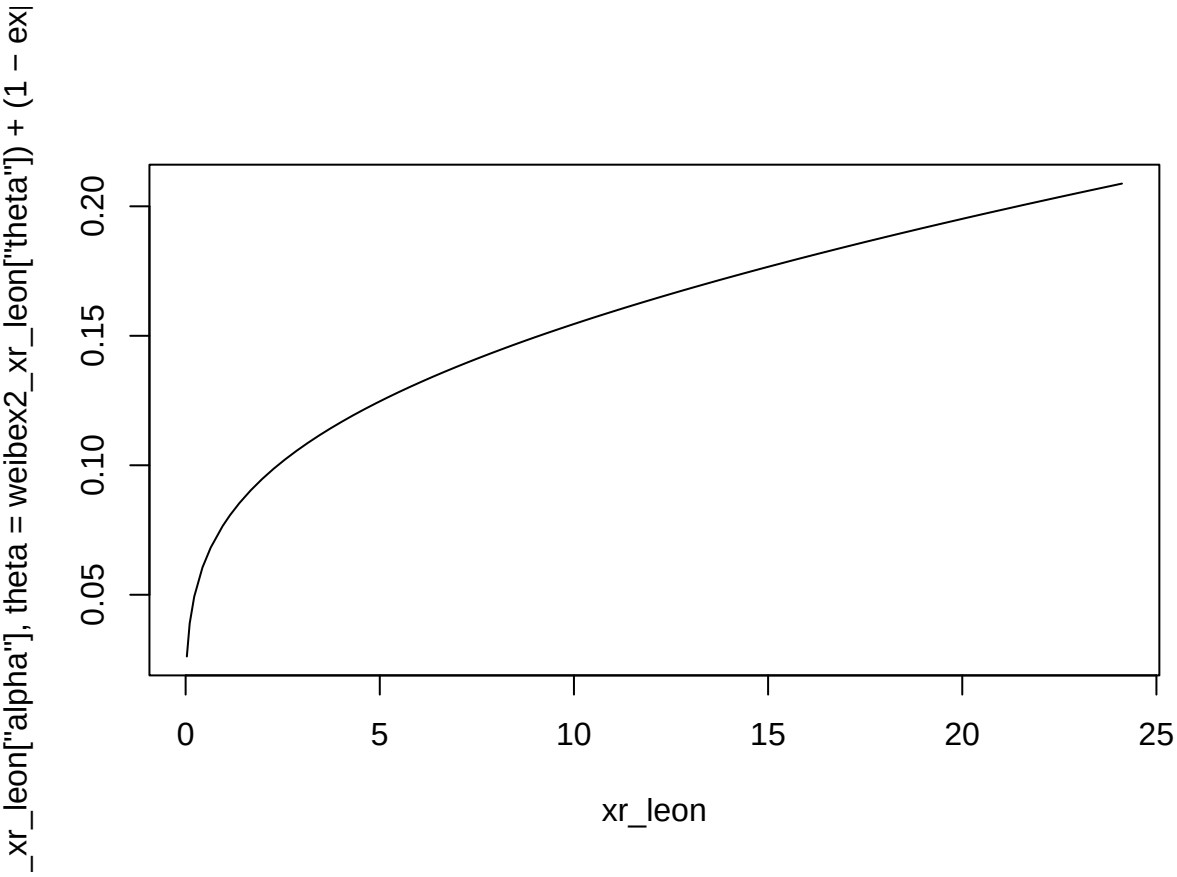
```
## $par
## [1] 0.115573530 4.115748320 0.003766985 0.001367109
##
## $value
## [1] 1.350252e-07
##
## $counts
## function gradient
##       55       55
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```



```
weibex2_xr_leon
```

```
##       alpha       theta        rate         mix
## 0.115573530 4.115748320 0.003766985 0.001367109
```

```
plot(xr_leon,
     (exp(weibex2_xr_leon["mix"]) / ( 1 + exp(weibex2_xr_leon["mix"])) *
        reliaR::pexpo.weibull(q = xr_leon,
                              alpha = weibex2_xr_leon["alpha"],
                              theta = weibex2_xr_leon["theta"]) +
        (1 - exp(weibex2_xr_leon["mix"]) / ( 1 + exp(weibex2_xr_leon["mix"]))) *
        stats::pexp(q = xr_leon,
                    rate = weibex2_xr_leon["rate"])),
     type = "l")
```

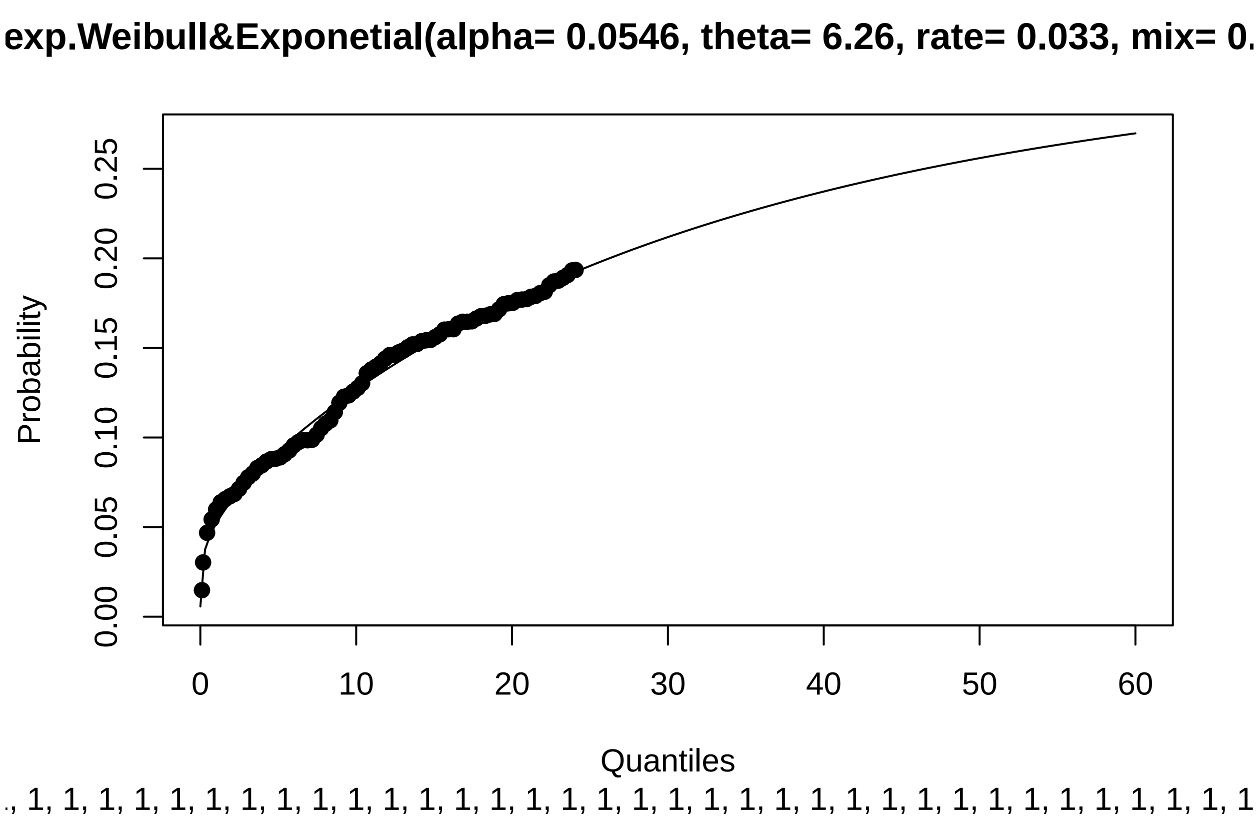



```
best_weibex2_xb_leon <- find_best_start_4parameter(p = yb_leon/100,
                                                   q = xb_leon,
                                                   max_shape = 1,
                                                   max_scale = 100,
                                                   max_rate = 0.7,
                                                   max_mix = 1,
                                                   steps_shape = 0.1,
                                                   steps_scale = 20,
                                                   steps_rate = 0.1,
                                                   steps_mix = 0.1,
                                                   fitting_function = get2weibex)
```

```
## Bester Startparameter: 0 0 0.6 0.9
## Bester Fehler: 2.144054e-07
```

```
weibex2_xb_leon <- get2weibex(
                        p = yb_leon/100,
                        q = xb_leon,
                        start = best_weibex2_xb_leon,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

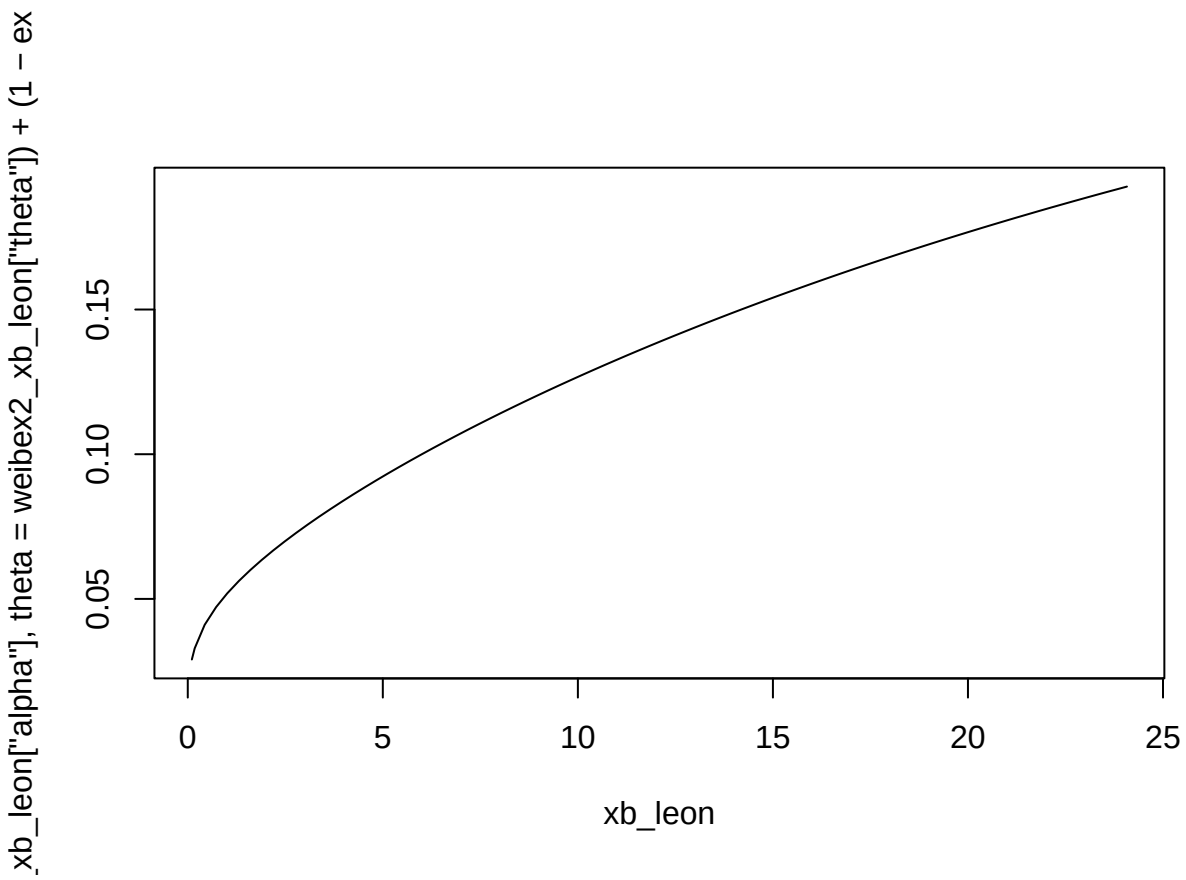
```
## $par
## [1] 0.05460768 6.26166552 0.03298447 1.38079932
##
## $value
## [1] 2.144054e-07
##
## $counts
## function gradient
##       20       20
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```



```
weibex2_xb_leon
```

```
##      alpha      theta       rate        mix
## 0.05460768 6.26166552 0.03298447 1.38079932
```

```
plot(xb_leon,
     (exp(weibex2_xb_leon["mix"]) / ( 1 + exp(weibex2_xb_leon["mix"])) *
        reliaR::pexpo.weibull(q = xb_leon,
                              alpha = weibex2_xb_leon["alpha"],
                              theta = weibex2_xb_leon["theta"]) +
        (1 - exp(weibex2_xb_leon["mix"]) / ( 1 + exp(weibex2_xb_leon["mix"]))) *
        stats::pexp(q = xb_leon,
                    rate = weibex2_xb_leon["rate"])),
     type = "l")
```



Exponentiierte Weibullverteilung ohne Lambda = 1

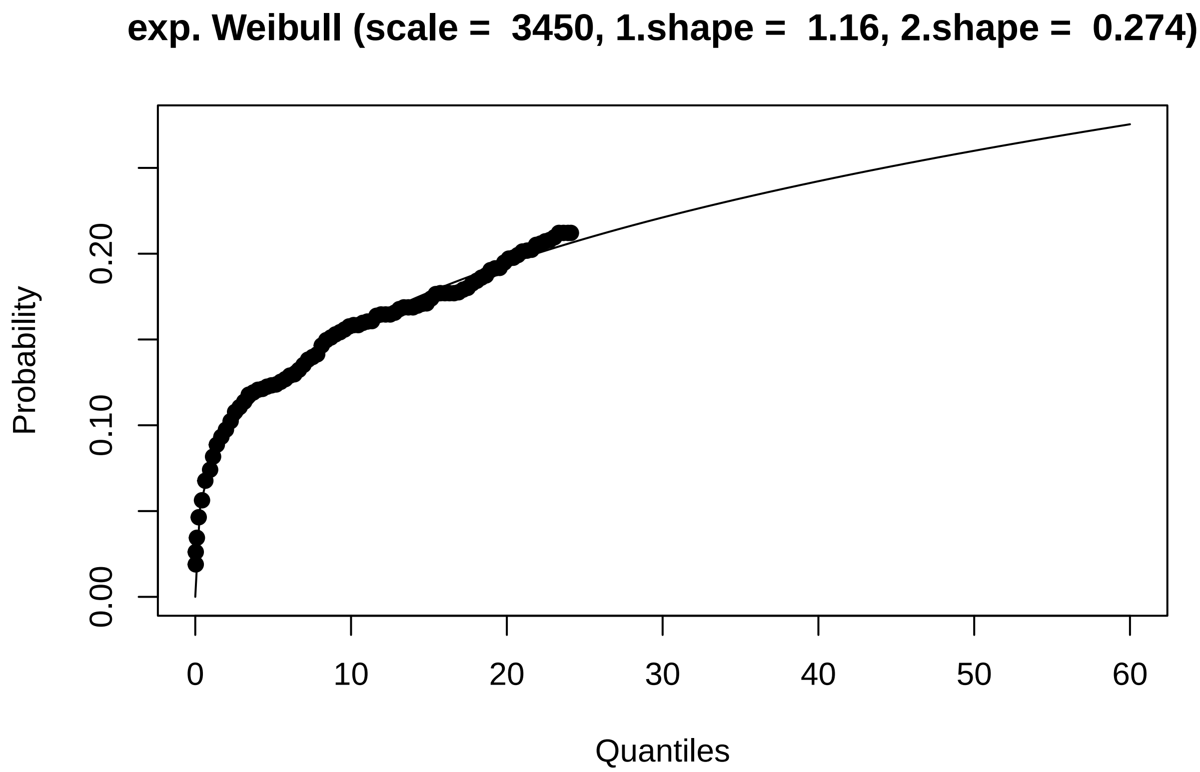
```
# exponentiierte Weibullverteilung ohne Lambda = 1
source("C:/Users/Chau/Documents/Masterarbeit/R Funktionen/getexpweib.R")

best_expweib_xr_leon <- find_best_start_3parameter(p = yr_leon/100,
                                                   q = xr_leon,
                                                   max_shape1 = 10,
                                                   max_shape2 = 10,
                                                   max_scale = 10,
                                                   steps_shape1 = 1,
```

```
                                                    steps_shape2 = 1,
                                                    steps_scale = 1,
                                                    fitting_function = getexpweib)
```

```
## Bester Startparameter: 1 9 6
## Bester Fehler: 1.819559e-07
```

```
expweib_xr_leon <- getexpweib(
                        p = yr_leon/100,
                        q = xr_leon,
                        start = best_expweib_xr_leon,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 3451.5728891    1.1582846    0.2744411
##
## $value
## [1] 1.819559e-07
##
## $counts
## function gradient
##       96       96
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

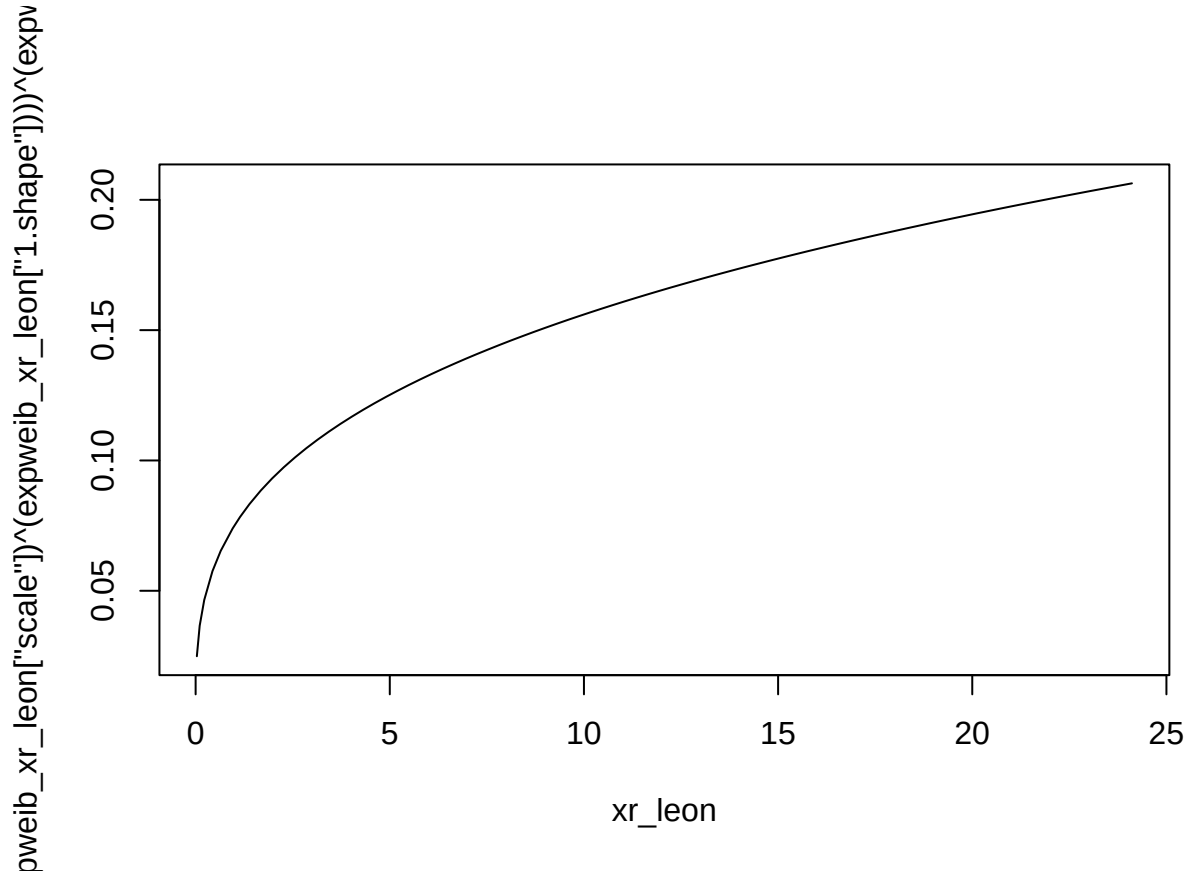


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
expweib_xr_leon
```

```
##        scale      1.shape      2.shape
## 3451.5728891    1.1582846    0.2744411
```

```
plot(xr_leon,
     (1 - exp(-(xr_leon / expweib_xr_leon["scale"])^
                     (expweib_xr_leon["1.shape"]))) ^ (expweib_xr_leon["2.shape"]),
     type = "l")
```



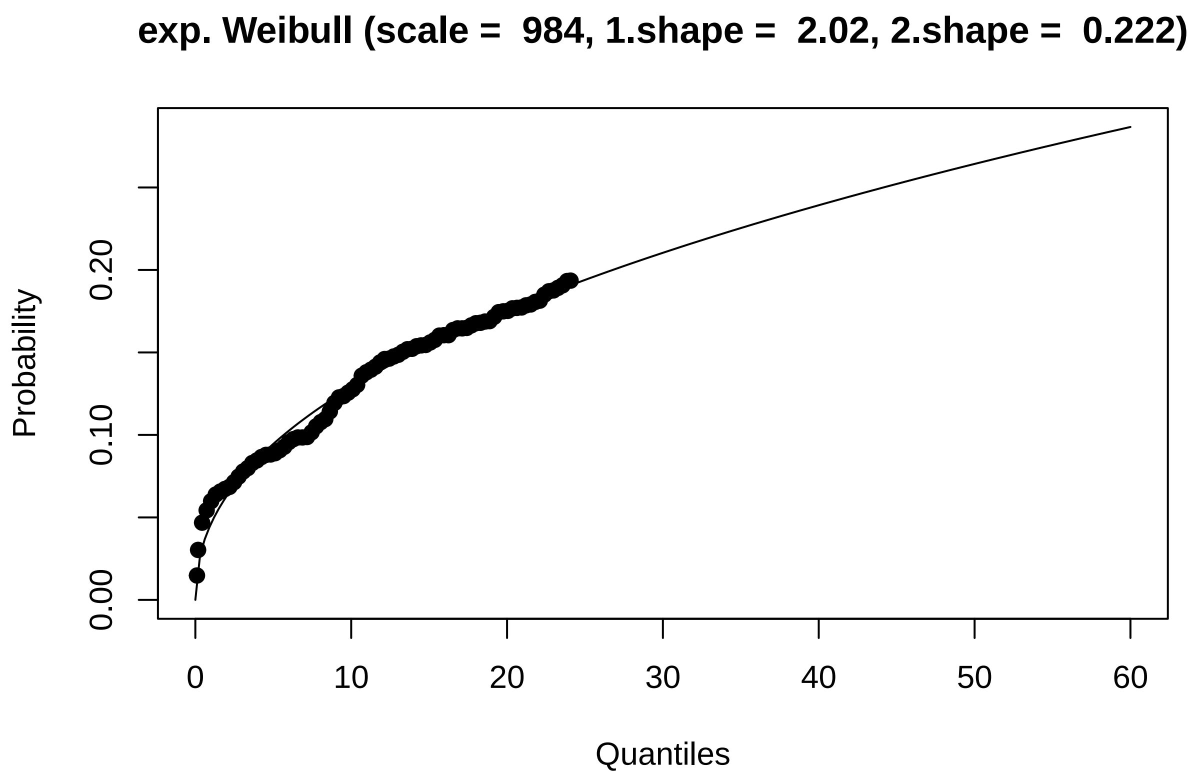

```
best_expweib_xb_leon <- find_best_start_3parameter(p = yb_leon/100,
                                                   q = xb_leon,
                                                   max_shape1 = 10,
                                                   max_shape2 = 10,
                                                   max_scale = 10,
                                                   steps_shape1 = 1,
                                                   steps_shape2 = 1,
                                                   steps_scale = 1,
                                                   fitting_function = getexpweib)
```

```
## Bester Startparameter: 1 9 3
## Bester Fehler: 3.329871e-07
```

```
expweib_xb_leon <- getexpweib(
                        p = yb_leon/100,
                        q = xb_leon,
                        start = best_expweib_xb_leon,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 983.5127552   2.0154744   0.2215924
##
## $value
## [1] 3.329871e-07
##
```

```
## $counts
## function gradient
##       65       65
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

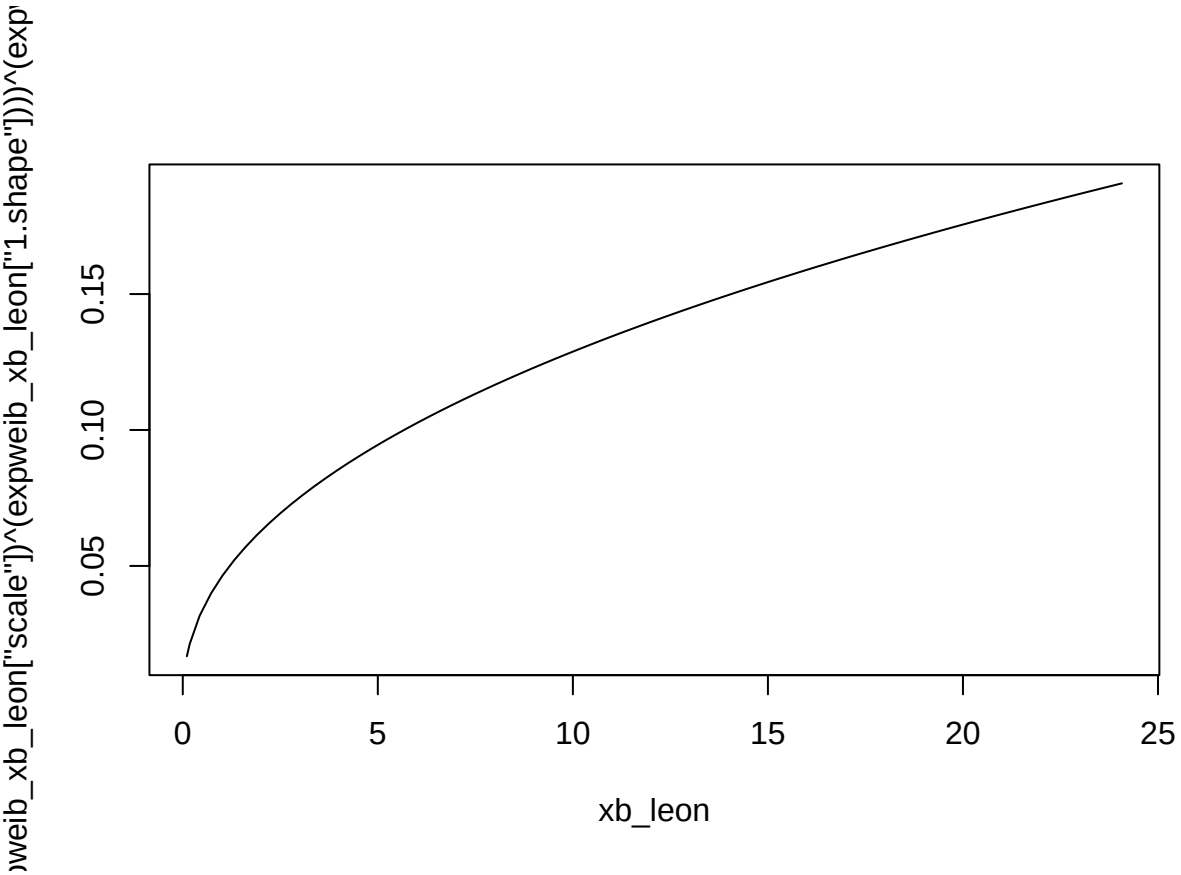


., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

```r
expweib_xb_leon
```

```
##       scale     1.shape     2.shape
## 983.5127552   2.0154744   0.2215924
```

```r
plot(xb_leon,
     (1 - exp(-(xb_leon / expweib_xb_leon["scale"])^(expweib_xb_leon["1.shape"])))^(expweib_xb_leon["2.s
     type = "l")
```



3-Stufige Exponetialverteilung

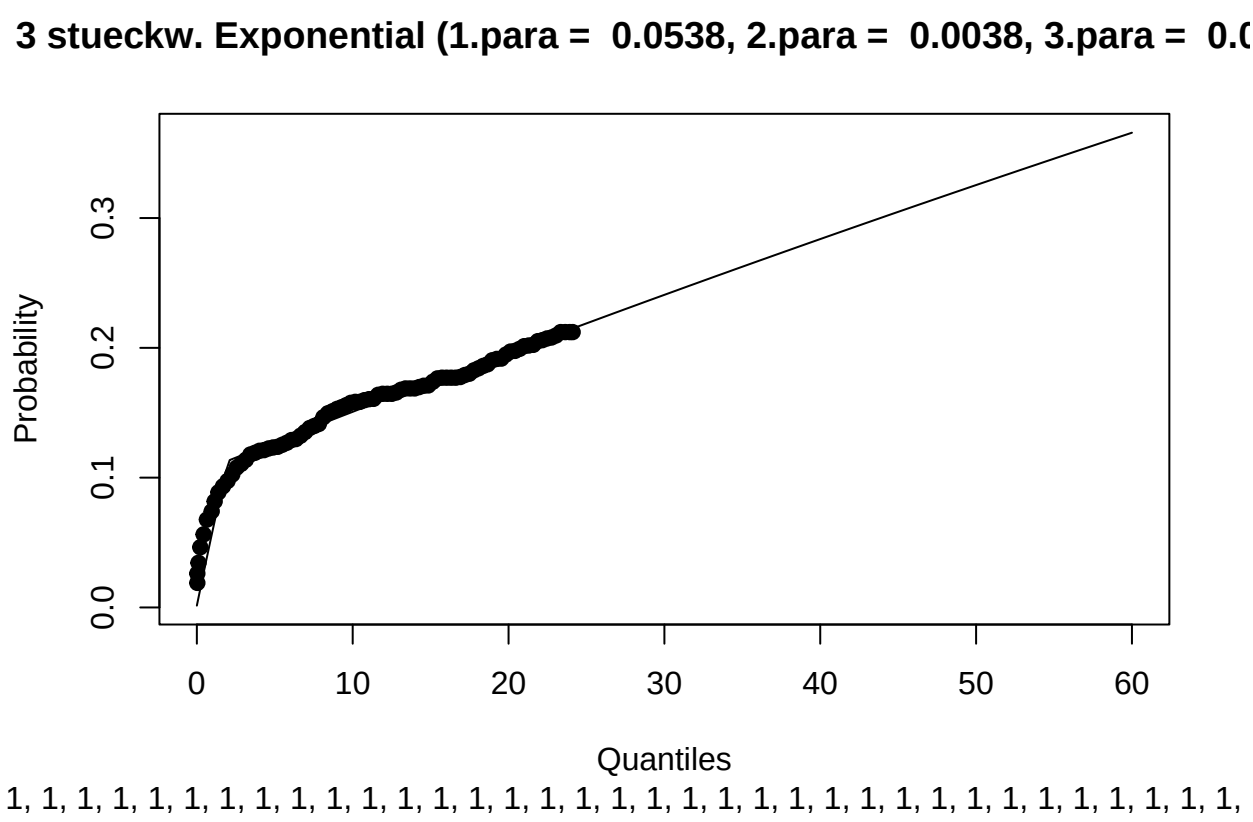
```
# 3-Stufige Exponetialverteilung
source("C:/Users/Chau/Documents/Masterarbeit/R Funktionen/getstuexp3.R")

best_stuexp3_xr_leon <- find_best_start_3parameter(p = yr_leon/100,
                                                   q = xr_leon,
                                                   max_shape1 = 0.1,
                                                   max_shape2 = 0.1,
                                                   max_scale = 0.1,
                                                   steps_shape1 = 0.01,
                                                   steps_shape2 = 0.01,
                                                   steps_scale = 0.01,
                                                   fitting_function = getstuexp3)
```

```
## Bester Startparameter: 0.05 0.1 0
## Bester Fehler: 8.199165e-07
```

```
stuexp3_xr_leon <- getstuexp3(
                        p = yr_leon/100,
                        q = xr_leon,
                        start = best_stuexp3_xr_leon,
                        show.output = TRUE,
                        plot = TRUE,
                        wert1 = 2,
                        wert2 = 6
```

```
                        )
```

```
## $par
## [1] 0.05380697 0.00380353 0.00100000
##
## $value
## [1] 8.199165e-07
##
## $counts
## function gradient
##       14       14
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

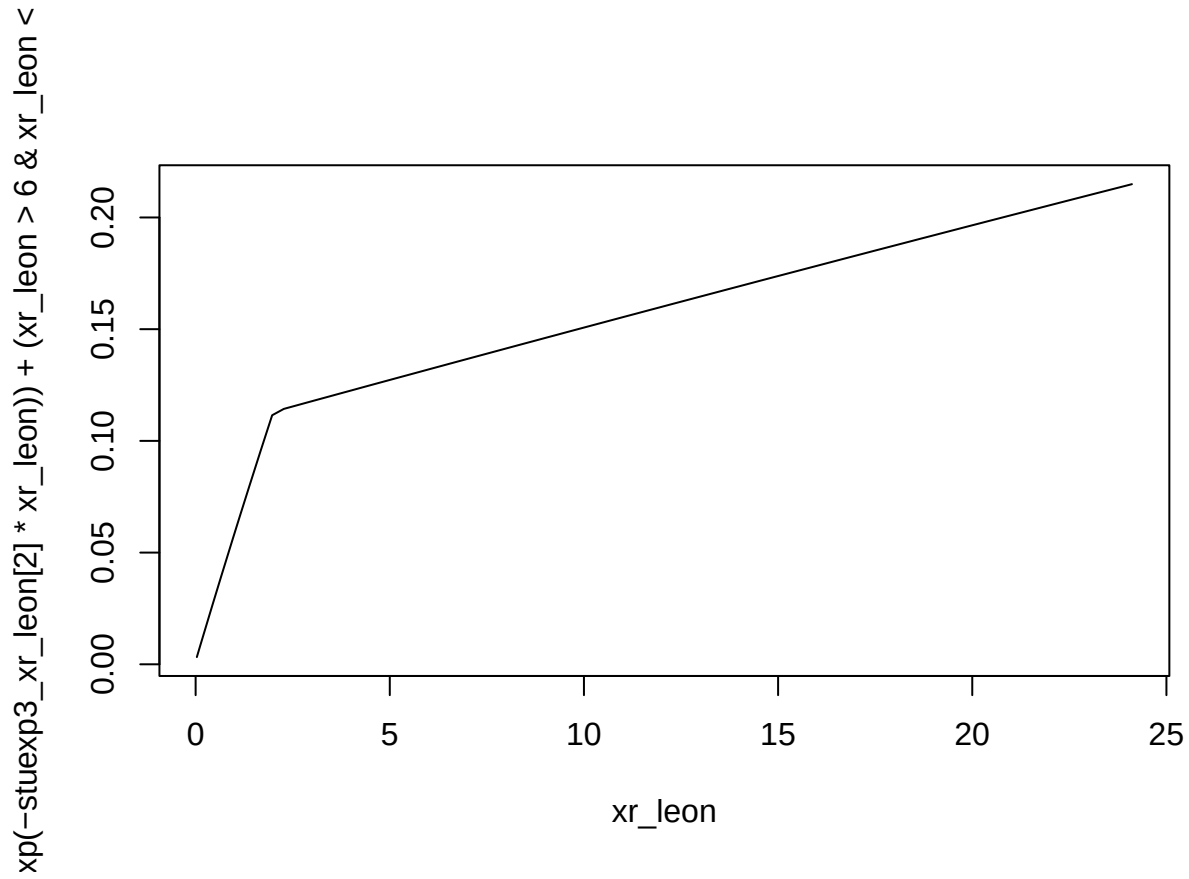



```
stuexp3_xr_leon
```

```
##     1.para     2.para     3.para
## 0.05380697 0.00380353 0.00100000
```

```
plot(xr_leon,
     ((xr_leon > 0 & xr_leon <= 2 ) * (1 - exp(-stuexp3_xr_leon[1] * xr_leon)) +
                                  (xr_leon > 2 & xr_leon <= 6 ) *
              (1 - exp(-stuexp3_xr_leon[1] * 2))  +
```

```
                                    (exp(-2 * stuexp3_xr_leon[2]) -
                                            exp(-stuexp3_xr_leon[2] * xr_leon)) +
                                    (xr_leon > 6 & xr_leon <= 65 ) *
            (1 - exp(-stuexp3_xr_leon[1] * 2)) +
                                    (exp(-2 * stuexp3_xr_leon[2]) - exp(-6 * stuexp3_xr_leon[2])) +
                                    (exp(-6 * stuexp3_xr_leon[3]) -
                                            exp(-stuexp3_xr_leon[3] * xr_leon))),
    type = "l")
```



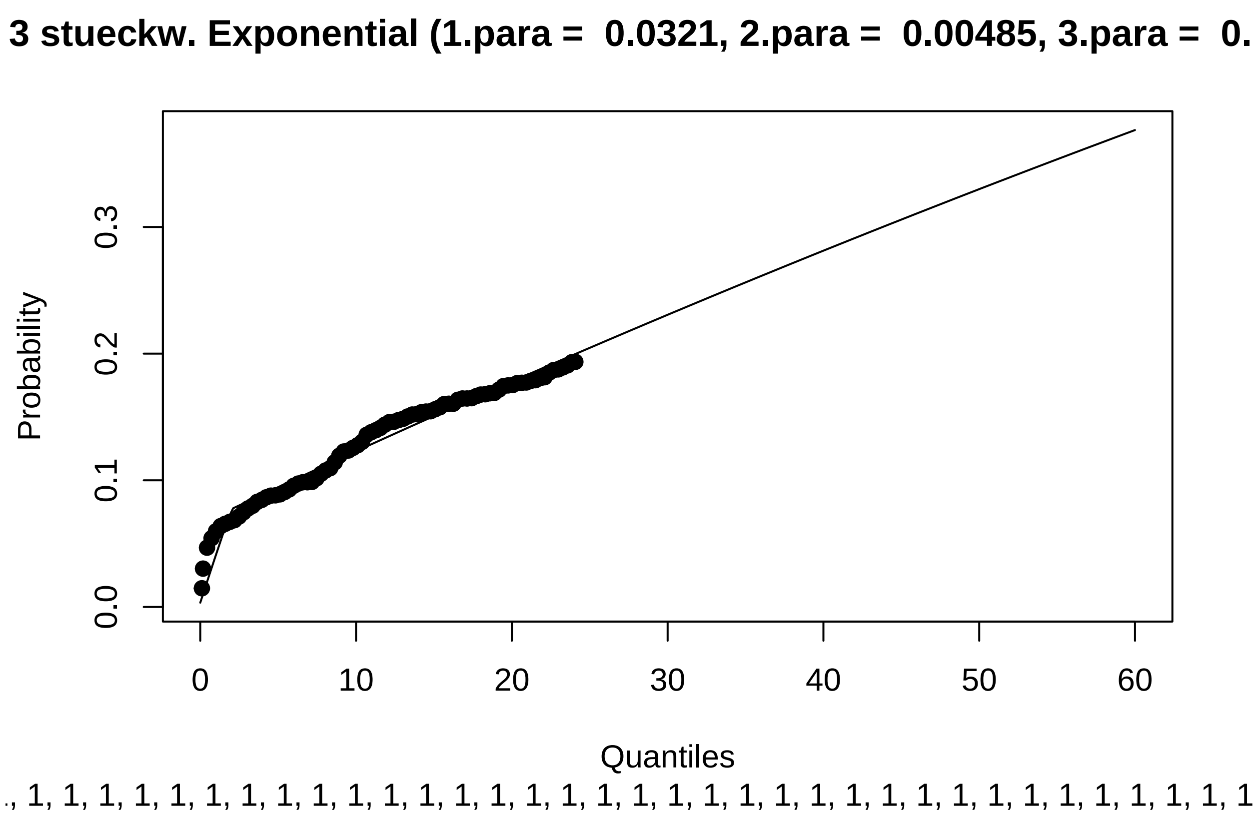

```
best_stuexp3_xb_leon <- find_best_start_3parameter(p = yb_leon/100,
                                                   q = xb_leon,
                                                   max_shape1 = 0.1,
                                                   max_shape2 = 0.1,
                                                   max_scale = 0.1,
                                                   steps_shape1 = 0.01,
                                                   steps_shape2 = 0.01,
                                                   steps_scale = 0.01,
                                                   fitting_function = getstuexp3)
```

```
## Bester Startparameter: 0.1 0.05 0.03
## Bester Fehler: 7.111325e-07
```

```
stuexp3_xb_leon <- getstuexp3(
                        p = yb_leon/100,
                        q = xb_leon,
                        start = best_stuexp3_xb_leon,
                        show.output = TRUE,
```

```
                         plot = TRUE,
                         wert1 = 2,
                         wert2 = 6
                       )
```

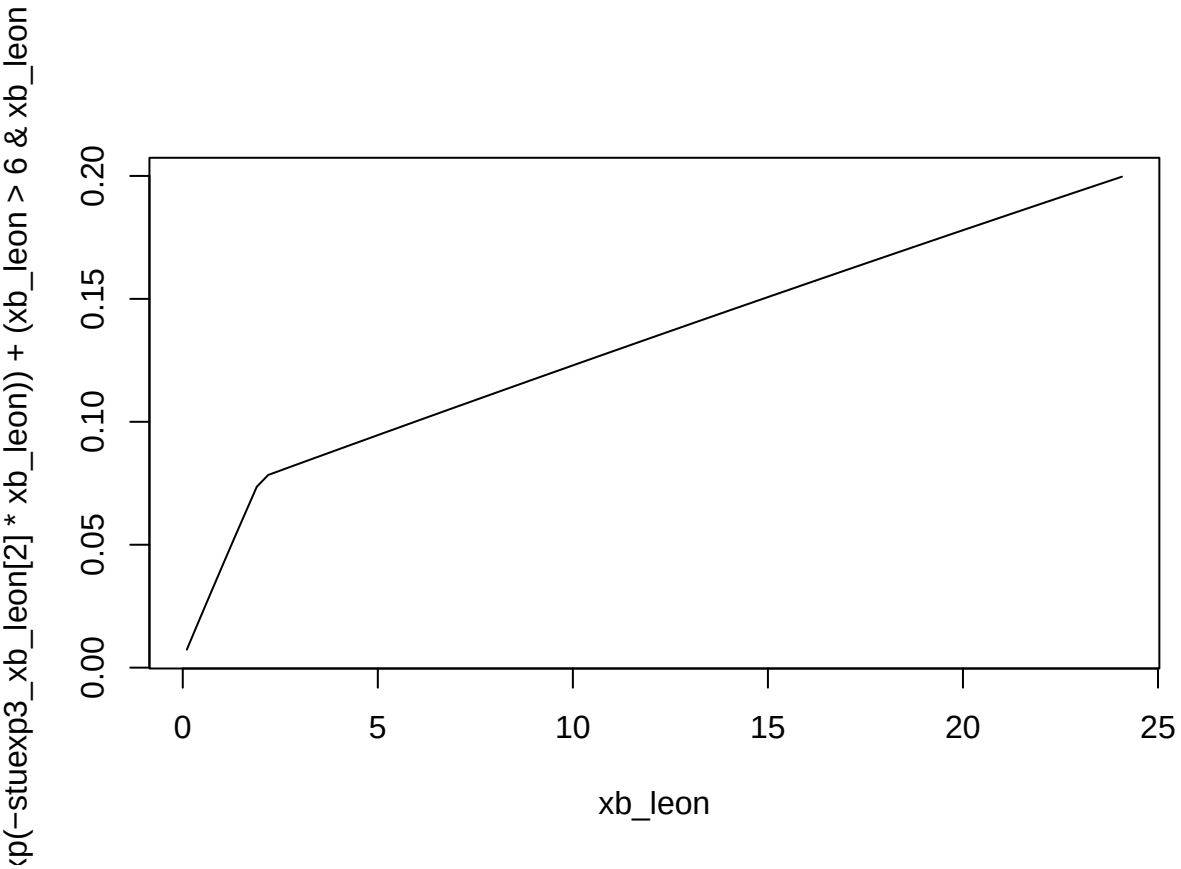
```
## $par
## [1] 0.032117721 0.004853818 0.001000000
##
## $value
## [1] 7.111325e-07
##
## $counts
## function gradient
##       18       18
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```




```
stuexp3_xb_leon
```

```
##      1.para      2.para      3.para
## 0.032117721 0.004853818 0.001000000
```

```r
plot(xb_leon,
     ((xb_leon > 0 & xb_leon <= 2 ) * (1 - exp(-stuexp3_xb_leon[1] * xb_leon)) +
                                    (xb_leon > 2 & xb_leon <= 6 ) *
              (1 - exp(-stuexp3_xb_leon[1] * 2))  +
                                    (exp(-2 * stuexp3_xb_leon[2]) - exp(-stuexp3_xb_leon[2] *
                                                                        xb_leon)) +
                                    (xb_leon > 6 & xb_leon <= 65 ) *
              (1 - exp(-stuexp3_xb_leon[1] * 2)) +
                                    (exp(-2 * stuexp3_xb_leon[2]) - exp(-6 * stuexp3_xb_leon[2])) +
                                    (exp(-6 * stuexp3_xb_leon[3]) -
                                             exp(-stuexp3_xb_leon[3] * xb_leon))),
     type = "l")
```



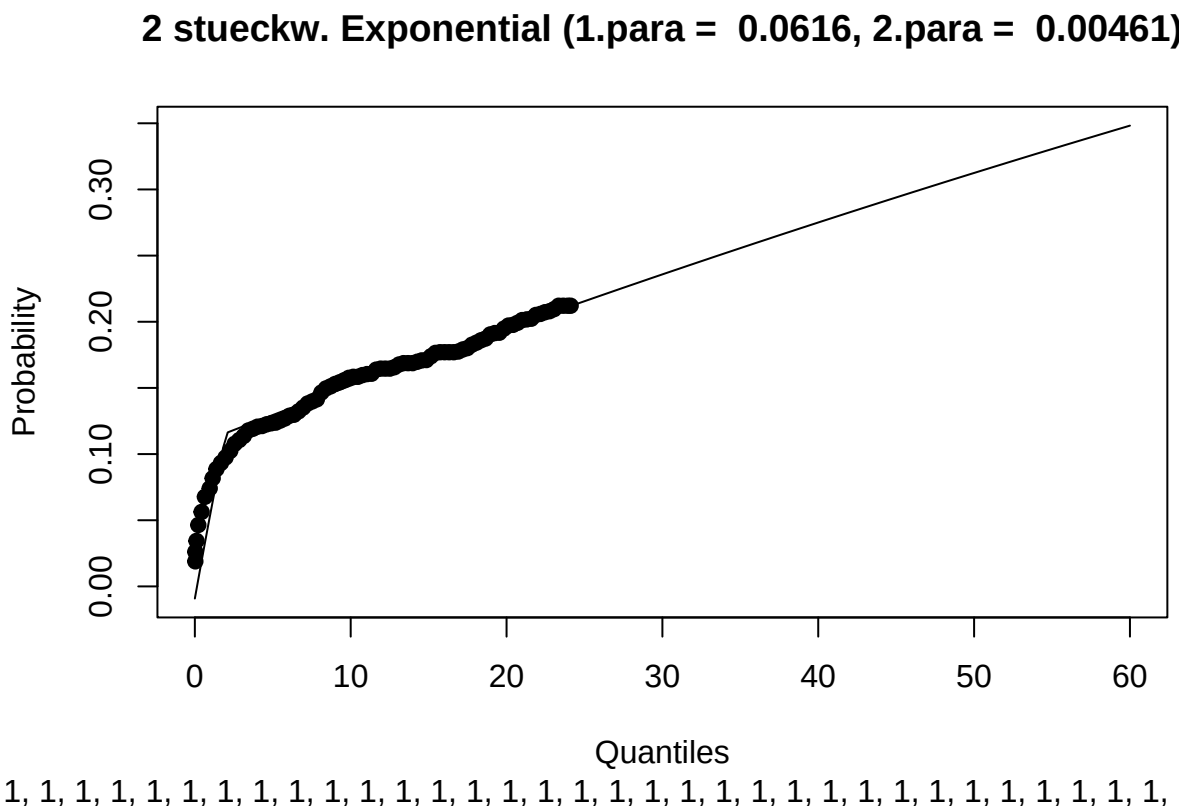
2-Stufige Exponetialverteilung

```r
# 2-Stufige Exponetialverteilung
source("C:/Users/Chau/Documents/Masterarbeit/R Funktionen/getstuexp2.R")

best_stuexp2_xr_leon <- find_best_start_2parameter(p = yr_leon/100,
                                                   q = xr_leon,
                                                   max_beta = 1,
                                                   max_eta = 0.5,
                                                   steps_beta = 0.1,
                                                   steps_eta = 0.01,
                                                   fitting_function = getstuexp2)
```

```
## Bester Startparameter: 0.8 0.14
## Bester Fehler: 1.313331e-06
```

```
stuexp2_xr_leon <- getstuexp2(
                        p = yr_leon/100,
                        q = xr_leon,
                        start = best_stuexp2_xr_leon,
                        show.output = TRUE,
                        plot = TRUE,
                        wert1 = 2
                      )
```

```
## $par
## [1] 0.061596745 0.004606699
##
## $value
## [1] 1.313331e-06
##
## $counts
## function gradient
##       21       21
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

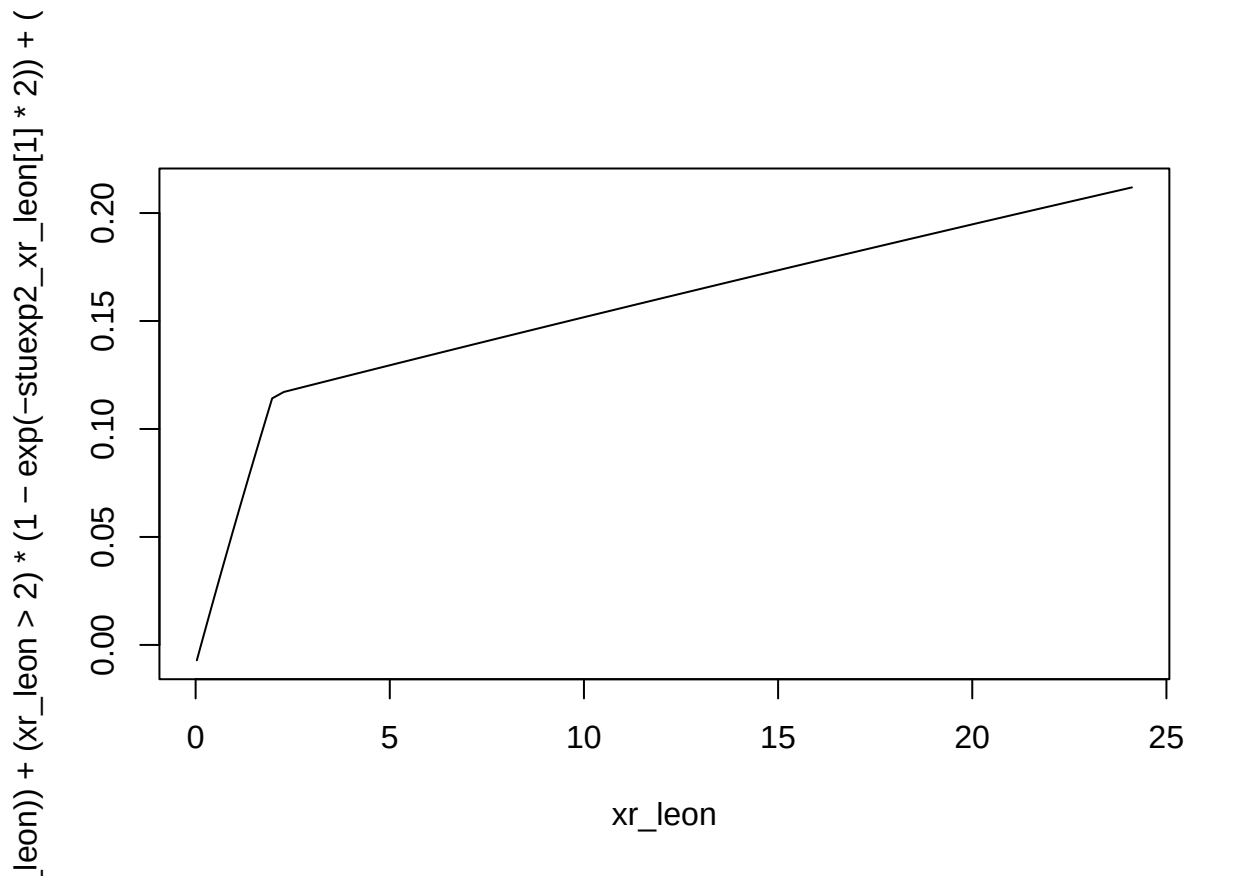


1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
stuexp2_xr_leon
```

```
##      1.para      2.para
## 0.061596745 0.004606699
```

```
plot(xr_leon,
     ((xr_leon > 0 & xr_leon <= 2 ) * (1 - exp(-stuexp2_xr_leon[1] * xr_leon)) +
                                     (xr_leon > 2 ) *  (1 - exp(-stuexp2_xr_leon[1] * 2))  +
                                     (exp(-2 * stuexp2_xr_leon[2]) -
                                              exp(-stuexp2_xr_leon[2] * xr_leon))),
     type = "l")
```



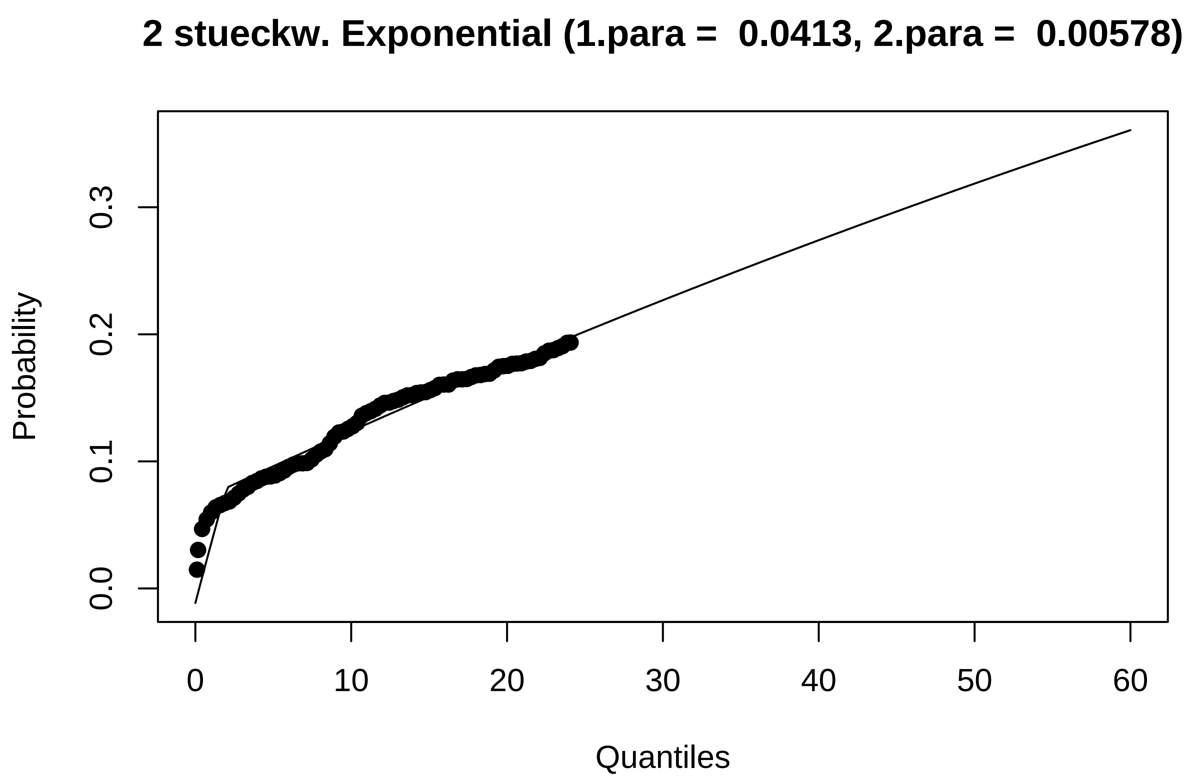

```
best_stuexp2_xb_leon <- find_best_start_2parameter(p = yb_leon/100,
                                                   q = xb_leon,
                                                   max_beta = 1,
                                                   max_eta = 0.5,
                                                   steps_beta = 0.1,
                                                   steps_eta = 0.01,
                                                   fitting_function = getstuexp2)
```

```
## Bester Startparameter: 0.1 0.14
## Bester Fehler: 1.100468e-06
```

```
stuexp2_xb_leon <- getstuexp2(
                        p = yb_leon/100,
                        q = xb_leon,
                        start = best_stuexp2_xb_leon,
```

```r
                            show.output = TRUE,
                            plot = TRUE,
                            wert1 = 2
                          )
```

```
## $par
## [1] 0.041253365 0.005777781
##
## $value
## [1] 1.100468e-06
##
## $counts
## function gradient
##       18       18
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

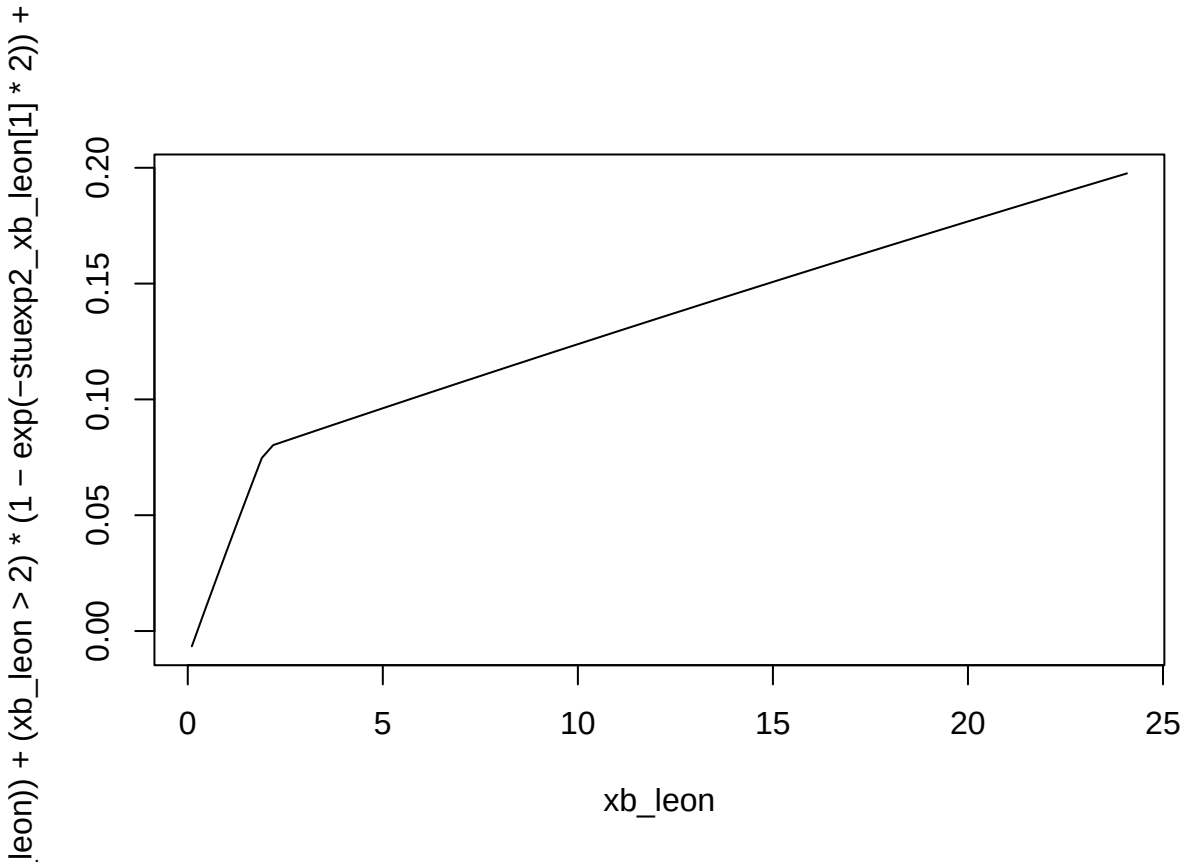


., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

```r
stuexp2_xb_leon
```

```
##      1.para      2.para
## 0.041253365 0.005777781
```

```r
plot(xb_leon,
     ((xb_leon > 0 & xb_leon <= 2 ) * (1 - exp(-stuexp2_xb_leon[1] * xb_leon)) +
                                     (xb_leon > 2 ) *  (1 - exp(-stuexp2_xb_leon[1] * 2))  +
                                     (exp(-2 * stuexp2_xb_leon[2]) -
                                              exp(-stuexp2_xb_leon[2] * xb_leon))),
     type = "l")
```



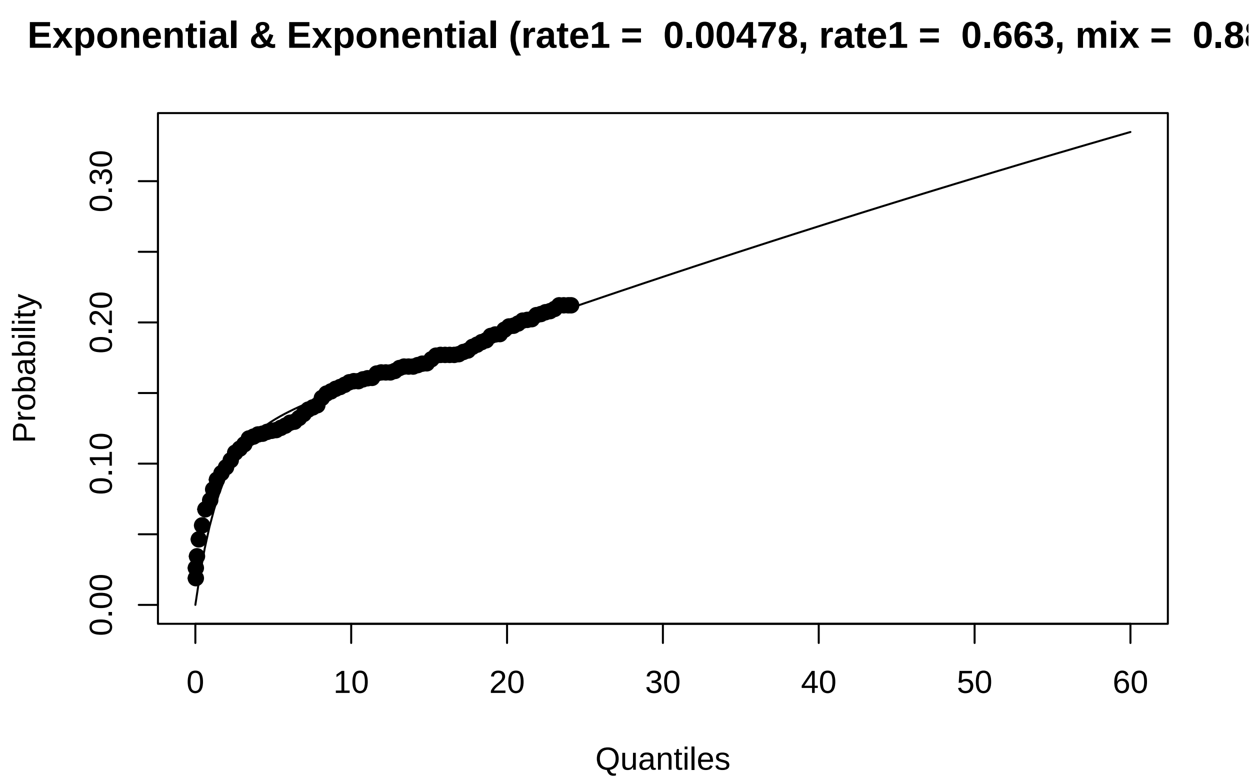
Mischung aus 2 Exponentialverteilungen

```r
# Mischung aus 2 Exponentialverteilungen
source("C:/Users/Chau/Documents/Masterarbeit/R Funktionen/getexex.R")

best_exex_xr_leon <- find_best_start_3parameter(p = yr_leon/100,
                                                q = xr_leon,
                                                max_shape1 = 0.7,
                                                max_shape2 = 0.7,
                                                max_scale = 1,
                                                steps_shape1 = 0.1,
                                                steps_shape2 = 0.1,
                                                steps_scale = 0.1,
                                                fitting_function = getexex)
```

```
## Bester Startparameter: 0.1 0.7 0.6
## Bester Fehler: 7.453511e-07
```

```r
exex_xr_leon <- getexex(
                        p = yr_leon/100,
                        q = xr_leon,
                        start = best_exex_xr_leon,
                        show.output = TRUE,
                        plot = TRUE
                      )
```

```
## $par
## [1] 0.0047809 0.6626663 2.0522614
##
## $value
## [1] 7.453511e-07
##
## $counts
## function gradient
##       16       16
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```



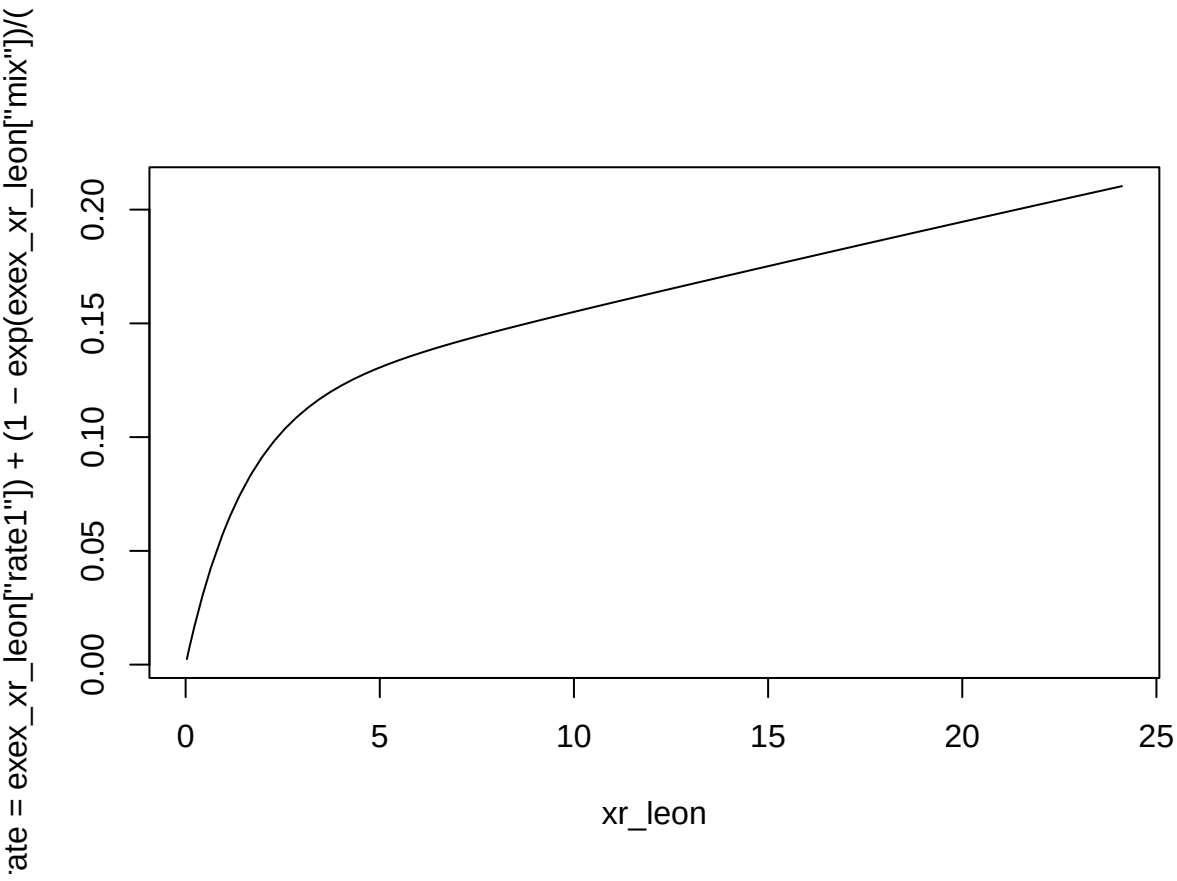
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```r
exex_xr_leon
```

```
##      rate1     rate2       mix
```

```
## 0.0047809 0.6626663 2.0522614
```

```
plot(xr_leon,
     (exp(exex_xr_leon["mix"]) / ( 1 + exp(exex_xr_leon["mix"])) *  stats::pexp(q = xr_leon,
                                                                                rate = exex_xr_leon["rate1"]
                      (1 - exp(exex_xr_leon["mix"]) / ( 1 + exp(exex_xr_leon["mix"]))) *
              stats::pexp(q = xr_leon,
                          rate = exex_xr_leon["rate2"])),
     type = "l")
```



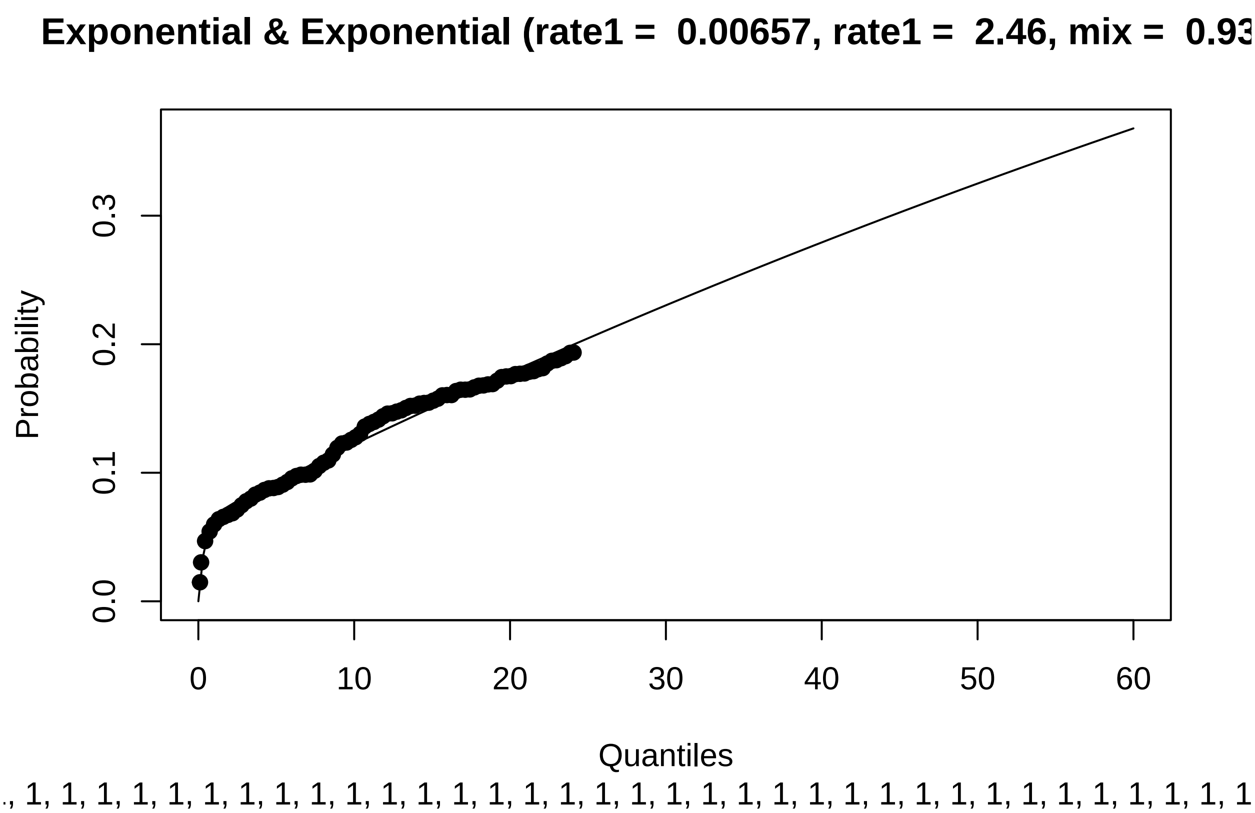

```
best_exex_xb_leon <- find_best_start_3parameter(p = yb_leon/100,
                                                q = xb_leon,
                                                max_shape1 = 0.7,
                                                max_shape2 = 0.7,
                                                max_scale = 1,
                                                steps_shape1 = 0.1,
                                                steps_shape2 = 0.1,
                                                steps_scale = 0.1,
                                                fitting_function = getexex)
```

```
## Bester Startparameter: 0.1 0.7 0.7
## Bester Fehler: 3.846759e-07
```

```
exex_xb_leon <- getexex(
                        p = yb_leon/100,
                        q = xb_leon,
                        start = best_exex_xb_leon,
                        show.output = TRUE,
```

```
                        plot = TRUE
                        )
```

```
## $par
## [1] 0.006567548 2.463084847 2.705563929
##
## $value
## [1] 3.846759e-07
##
## $counts
## function gradient
##       31       31
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

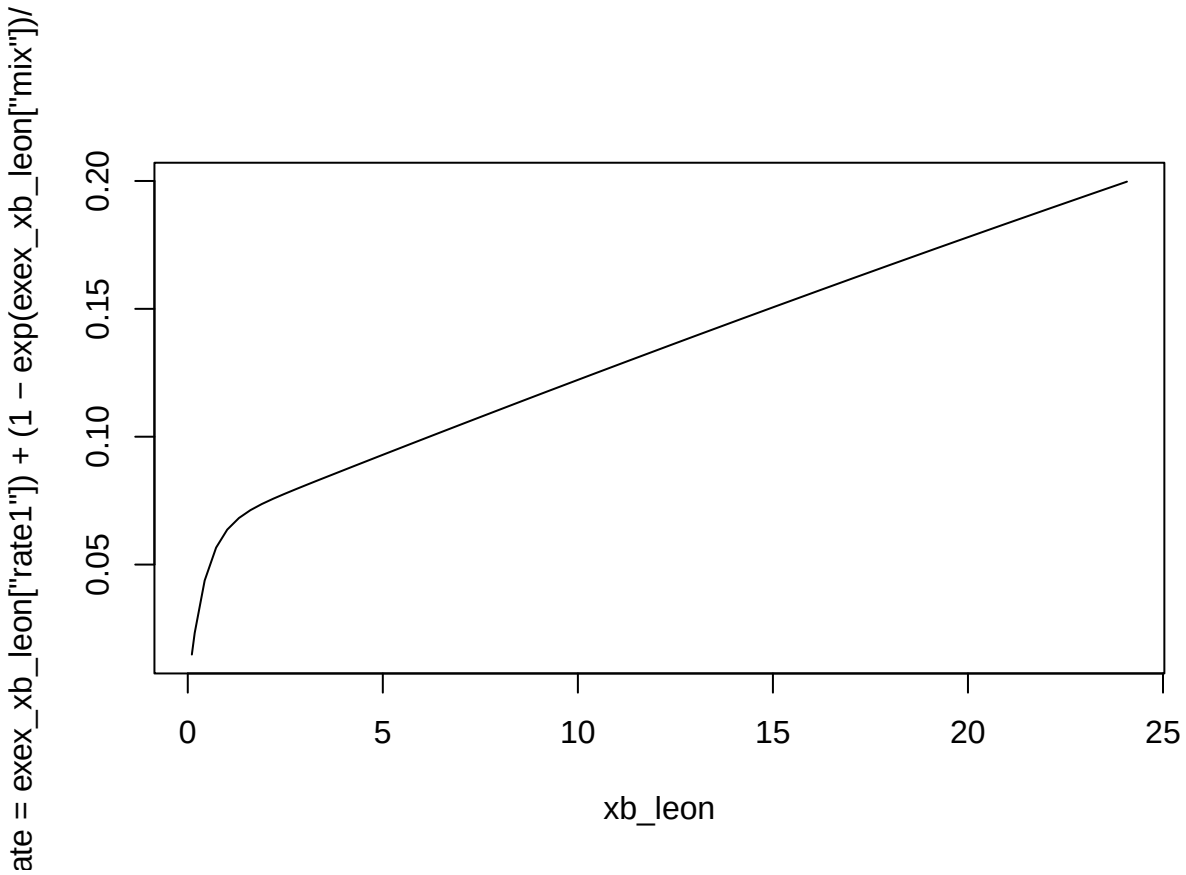



```
exex_xb_leon
```

```
##       rate1       rate2         mix
## 0.006567548 2.463084847 2.705563929
```

```
plot(xb_leon,
     (exp(exex_xb_leon["mix"]) / ( 1 + exp(exex_xb_leon["mix"])) *  stats::pexp(q = xb_leon,
                                                                            rate = exex_xb_leon["rate1"]
```

```
                    (1 - exp(exex_xb_leon["mix"]) / ( 1 + exp(exex_xb_leon["mix"]))) *
            stats::pexp(q = xb_leon,
                        rate = exex_xb_leon["rate2"])),
    type = "l")
```



Ergebnnis

```
getvalue <- function(p, q, best_start, fitting_function){
  if(identical(as.character(substitute(fitting_function)), "getstuexp2")){
    output <- capture.output({
    result <- getstuexp2(p = p, q = q, start = best_start, show.output = TRUE,
                         plot = FALSE, wert1 = 2)
    })
  }
  else if(identical(as.character(substitute(fitting_function)), "getstuexp3")){
          output <- capture.output({
            result <- getstuexp3(
                          p = p, q = q, start = best_start,
                          show.output = TRUE, plot = FALSE, wert1 = 2, wert2 = 6)
          })
  }
  else{
    output <- capture.output({
    result <- fitting_function(p = p, q = q, start = best_start,
```

```
                                show.output = TRUE, plot = FALSE)
    })
  }

  # Berechne den Fehler (leon_r_1$value) für die aktuellen Startparameter
  error <- as.numeric(gsub("\\[1\\]\\s+", "", output[5]))

  return(error)
}

best_test <- function(p, q, weibull, weib, weibex, weibex2, expweib, stuexp3, stuexp2,
                      exex, start_weibull, start_weib, start_weibex, start_weibex2,
                      start_expweib, start_stuexp3, start_stuexp2, start_exex,
                      group){
  weibull_val <- getvalue(p, q, start_weibull, getweibullpar)
  weib_val <- getvalue(p, q, start_weib, getweibpar)
  weibex_val <- getvalue(p, q, start_weibex, getweibex)
  weibex2_val <- getvalue(p, q, start_weibex2, get2weibex)
  expweib_val <- getvalue(p, q, start_expweib, getexpweib)
  stuexp3_val <- getvalue(p, q, start_stuexp3, getstuexp3)
  stuexp2_val <- getvalue(p, q, start_stuexp2, getstuexp2)
  exex_val <- getvalue(p, q, start_exex, getexex)

  error_distribution_pairs <- list(
    list(weibull_val, "W"),
    list(weib_val, "e.W."),
    list(weibex_val, "M. W&E"),
    list(weibex2_val, "M. e.W&E"),
    list(expweib_val, "e.W o. lambda = 1"),
    list(stuexp3_val, "3 s.E."),
    list(stuexp2_val, "2 s.E."),
    list(exex_val, "M. E&E")
  )

  # Suchen Verteilung mit dem kleinsten Fehler
  best_pair <- error_distribution_pairs[[which.min(sapply(
    error_distribution_pairs, function(pair) pair[[1]]))]]

  # Drucken Sie die Ergebnisse
  cat("Beste Verteilung:", best_pair[[2]], "\n")
  cat("Bester Fehler:", best_pair[[1]], "\n")
  cat("Gruppe: ", group)

  return(c(group, best_pair[[2]],  best_pair[[1]]))
}

best_tavr <- best_test(yb_leon/100, xb_leon, leon_b_1, weibbull_xb_leon,
                       weibex_xb_leon, weibex2_xb_leon, expweib_xb_leon,
                       stuexp3_xb_leon, stuexp2_xb_leon, exex_xb_leon,
                       best_leon_b_1, best_weibbull_xb_leon,
                       best_weibex_xb_leon, best_weibex2_xb_leon,
                       best_expweib_xb_leon, best_stuexp3_xb_leon,
                       best_stuexp2_xb_leon, best_exex_xb_leon, "TAVR")
```

```
## Beste Verteilung: M. e.W&E
## Bester Fehler: 2.144054e-07
## Gruppe:  TAVR
```

```
best_savr <- best_test(yr_leon/100, xr_leon, leon_r_1, weibbull_xr_leon,
                       weibex_xr_leon, weibex2_xr_leon, expweib_xr_leon,
                       stuexp3_xr_leon, stuexp2_xr_leon, exex_xr_leon,
                       best_leon_r_1, best_weibbull_xr_leon,
                       best_weibex_xr_leon, best_weibex2_xr_leon,
                       best_expweib_xr_leon, best_stuexp3_xr_leon,
                       best_stuexp2_xr_leon, best_exex_xr_leon, "SAVR")
```

```
## Beste Verteilung: M. e.W&E
## Bester Fehler: 1.350252e-07
## Gruppe:  SAVR
```

```
tab <- matrix(c("PARTNER2", "MiRi", "TSmF", best_tavr[1], best_tavr[2],
                best_tavr[3], weibex2_xb_leon[1:3], NA, NA, NA,
                weibex2_xb_leon[4],
                "PARTNER2", "MiRi", "TSmF", best_tavr[1], "M. W&E",
                getvalue(yb_leon/100, xb_leon, best_weibex_xb_leon, getweibex),
                NA, NA, weibex_xb_leon[1:2], NA, weibex_xb_leon[3:4],
                "PARTNER2", "MiRi", "TSmF",  best_savr[1], best_savr[2],
                best_savr[3], weibex2_xr_leon[1:3], NA, NA, NA,
                weibex2_xr_leon[4],
                "PARTNER2", "MiRi", "TSmF",  best_savr[1], "M. W&E",
                getvalue(yr_leon/100, xr_leon, best_weibex_xr_leon, getweibex),
                NA, NA, weibex_xr_leon[1:2], NA, weibex_xr_leon[3:4]),
              ncol=13, byrow=TRUE)

rownames(tab) <- NULL
colnames(tab) <- c('Studie', 'PG', 'EP', 'GR', 'Verteilung', 'SSE', '$\\alpha$',
                   '$\\theta$', '$\\lambda_1$', '$\\lambda_2$', '$\\lambda_3$',
                   '$\\vartheta$', '$\\psi$')

results <- as.data.frame(tab)

# Speichern
write.table(results, "results_leon.txt", sep = "\t", row.names = FALSE)

# Funktion zur Überprüfung von NA-Werten für Zeichenketten und numerische Werte
is_non_empty <- function(x) {
  return(!is.na(x) & x != "")
}

# Spalten mit mindestens einem nicht-NA-Wert ermitteln
nicht_leere_spalten <- colSums(sapply(results, is_non_empty)) > 0

# Konvertieren Sie die Tabelle in eine Markdown-Tabelle
print(results[, nicht_leere_spalten])
```

```
##     Studie   PG   EP   GR Verteilung          SSE          $\\alpha$
## 1 PARTNER2 MiRi TSmF TAVR   M. e.W&E 2.144054e-07 0.0546076810089096
## 2 PARTNER2 MiRi TSmF TAVR     M. W&E 2.427772e-07               <NA>
## 3 PARTNER2 MiRi TSmF SAVR   M. e.W&E 1.350252e-07  0.115573530149325
```

```
## 4 PARTNER2 MiRi TSmF SAVR     M. W&E 1.869176e-07               <NA>
##          $\\theta$        $\\lambda_1$     $\\lambda_2$        $\\vartheta$
## 1 6.26166552478436  0.0329844661270292             <NA>                <NA>
## 2             <NA>   0.291473845635576 2495.12401615442 0.00732720814217303
## 3 4.11574831993765 0.00376698541182793             <NA>                <NA>
## 4             <NA>   0.347917968724667 197.750555554887               0.001
##               $\\psi$
## 1    1.38079932459548
## 2 0.00108717392639213
## 3 0.00136710915815876
## 4  0.0308439983698216
```